DOCUMENT RESUME

ED 228 050 SE 040 926

TITLE	Computers in the Mathematics Classroom. Guidelines for Quality Mathematics Teaching. Monograph No. 17.
INSTITUTION	Michigan Council of Teachers of Mathematics.
PUB DATE	Aug 82
NOTE	149p.
AVAILABLE FROM	MCTM, Box 16124, Lansing, MI 48901 ($5.00).
PUB TYPE	Guides - Classroom Use - Guides (For Teachers) (052)
EDRS PRICE	MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS	*Computer Oriented Programs; *Computer Programs; Computers; Elementary Secondary Education; Mathematical Concepts; *Mathematics Curriculum; *Mathematics Education; *Mathematics Instruction; Mathematics Skills; *Microcomputers; Programing
IDENTIFIERS	Michigan

ABSTRACT

This monograph is intended to be a practical introduction to computers for K-12 mathematics teachers, describing and illustrating ways in which computer technology can be used in the teaching of mathematics. It is organized according to the major ways computers are used in classrooms, and includes activities with suggested lesson outlines, BASIC program listings, and a brief glossary. Chapter 1 provides an introduction to computer-enhanced mathematics, with classroom applications in two broad categories considered: (1) computer science, in which the computer is the object of instruction; and (2) instructional computing, in which the computer is used for teaching and learning. The following seven chapters focus on computer literacy; teaching BASIC programming; programming activities for developing concepts; computer methods for problem solving; drill, practice, and tutorial uses; other uses; and resources for educational computing. Programs included are designed to run on most types of microcomputers; for this reason, they do not involve graphic or other features particular to one machine. All programs were run and listed on an Apple II microcomputer, and the collection can easily be stored on a single floppy disk. (MNS)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

THE MICHIGAN COUNCIL OF TEACHERS OF MATHEMATICS

Guidelines for
Quality Mathematics Teaching
A Monograph Series

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
✓ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

ED228030

COMPUTERS IN THE MATHEMATICS CLASSROOM

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY
Louis Henkel
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

SE040926

MONOGRAPH NO. 17
AUGUST 1982

Additional monographs may be obtained by writing (prepaid) to:

MCTM
Box 16124
Lansing, MI 48901

Make checks payable to Michigan Council of Teachers of Mathematics (M.C.T.M.)

Monographs available:

3. Order in Number *($1.00)*
4. Metric Measurement Activity Cards *($2.00)*
6. Michigan Fourth and Seventh Grade Student Performance in Mathematics 1970 - 1973 *($1.50)*
9. Activities in Elementary Probability *($1.00)*
10. Motion in Geometry *($3.00)*
11. Activities in Metric Measurement *($2.50)*
12. Uses of the Calculator in School Mathematics, K-12 *($1.50)*
13. Investigations in Mathematics *($2.00)*
14. High School Mathematics Preparation in Michigan: Assets and Liabilities *($1.50)*
15. Developing Reasoning Skills with Attribute Blocks *($2.50)*
16. A Curriculum in Geometry for Grades K-9 *($3.00)*
17. Computers in the Mathematics Classroom *($5.00)*

This monograph should not be construed to represent the official position of the Michigan Council of Teachers of Mathematics or its Executive Committee. It reflects the ideas and opinions of the authors, and is a substantial contribution to mathematics education.

©1982 Michigan Council
of Teachers of Mathematics

PREFACE

This monograph was outlined at a meeting in Albion, Michigan, in November of 1980. John Camp, Dwayne Channel, Carol Cody, Art Coxford, Art Davidge, Chris Hirsch, Wayne Kladder, Gary Marchionini, Al Monetta, and Betty Shaw attended that organizational meeting to share ideas, develop the structure for the monograph, and assume tasks.

The monograph is meant to be a practical introduction to computers for K-12 mathematics teachers. It is organized according to the major ways computers are used in classrooms, and includes: activities with suggested lesson outlines, BASIC program listings, a brief glossary, and many references and sources for further information. The programs are designed to run on most types of microcomputer. For this reason, programs included do not involve graphic or other features of microcomputers which are particular to each machine. All the programs were run and listed on an Apple II microcomputer, and the collection can easily be stored on a single floppy disk.

Many people played important roles in bringing this monograph to fruition. The authors spent many hours both in discussion and writing. Art Coxford contributed much time and effort in shaping the document. The MCTM Guidelines Committee members spent many hours reading and discussing the monograph and made many helpful suggestions. David Johnson was responsible for seeing the monograph through the final revisions and for its printing.

Gary Marchionini

TABLE OF CONTENTS

Chapter		Page
	Preface	
1	Introduction to Computer Enhanced Mathematics.....1 John Camp and Gary Marchionini	
2	Computer Literacy.....13 Al Monetta and Betty Shaw	
3	Teaching BASIC Programming.....22 Carol Cody and Betty Shaw	
4	Programming Activities for Developing Concepts.....41 Rheta Rubenstein	
5	Computer Methods for Problem Solving.....65 Dwayne Channell, Carol Cody and Christian Hirsch	
6	Drill, Practice and Tutorial Uses of Computers.....108 Art Davidge	
7	Other Uses of Computers.....118 Gary Marchionini and Al Monetta	
8	Resources for Educational Computing.....131 Wayne Kladder	

CHAPTER ONE

INTRODUCTION TO COMPUTER ENHANCED MATHEMATICS

JOHN CAMP
Wayne State University
Detroit, Michigan

GARY MARCHIONINI
Wayne State University
Detroit, Michigan

Teachers of mathematics have traditionally assumed leadership positions in educational applications of computers. From the early days of time-sharing systems, mathematics teachers have been the primary users of computers in classrooms. The recent developments in low-cost, stand- alone microcomputers have led to increased interest in computers in education, and mathematics teachers and their professional organizations have accepted increased responsibility for the growing applications of all types of computers in schools. The National Council of Teachers of Mathematics (NCTM, 1980) and National Council of Supervisors of Mathematics (NCSM, 1977) have called upon mathematics educators to expand their uses of computer technology.

The purpose of this monograph is to describe and illustrate ways in which computer technology can be used in the teaching of mathematics. If teachers are to make optimal use of computers in their classrooms, a comprehensive plan of action must be formulated. Software and hardware must be evaluated, selected, maintained, housed and updated. A variety of inservice provisions should be made, and a management scheme for sharing hardware and software among teachers and students must be devised. Decisions regarding these concerns are

dependent on curriculum goals and how computers are to be used to help achieve these goals. Because particular applications require certain hardware, software, inservice and management configurations, selecting appropriate applications for a given classroom must be an early decision in planning for using computers.

Classroom applications can be divided into two broad categories: computer science, and instructional computing. COMPUTER SCIENCE is a subject that deals with the study of computers - the computer is the object of instruction. INSTRUCTIONAL COMPUTING is a general phrase that includes any application of a computer in teaching and learning. Whether termed computer-enhanced, -assisted, -related, -based, or -aided, instructional computing represents those computer applications that help teachers teach, and students learn. Instructional computing applications focus on any subject matter (e.g. mathematics) as the object of instruction. The following two sections describe computer science applications and instructional computing applications respectively.

COMPUTER SCIENCE APPLICATIONS

Computer Literacy

Many definitions of computer literacy exist. Common characteristics include: hands-on hardware and software experiences, recognition of the uses and limitations of computers, and an awareness of the social impact of information processing technologies.

Although there is wide agreement on the necessity for providing computer literacy experiences for all students, there is not total agreement as to whether particular curriculum areas should be responsible for the training or if computer literacy should be infused throughout all curriculum areas. The social issues involved have prompted some social studies teachers to include computer literacy modules, and science teachers often cooperate with mathematics teachers in providing computer training. Computer literacy is considered in Chapter Two.

Computer Programming

Students who are able to command the computer through programming have a powerful tool for concept exploration, experimentation, and problem solving. Just as calculators have begun to affect the way teachers present some mathematical topics and how students learn and practice these topics, so computers will affect the teaching and learning of mathematics. The ability to program the computer can facilitate the introduction of content not traditionally included in the curriculum, as well as provide alternate approaches for presenting and learning existing content.

Like computer literacy, teaching of computer programming has traditionally been done by mathematics teachers. Although some educators have advocated separate computer science curricula, it is likely that mathematics teachers will continue to provide the bulk of computer programming training in schools. Chapter Three focuses on teaching BASIC language programming.

INSTRUCTIONAL COMPUTING APPLICATIONS

The simplest way to classify instructional computing applications is by programming and programs. In the first case, programming exercises designed to promote the learning of mathematics content are assigned. In the second case, programs (drill and practice, simulation, management, etc.) are available for students and teachers to use.

Developing and Reinforcing Concepts Through Programming

Writing computer programs to solve mathematical exercises is analogous to "teaching the computer". Students who write computer programs to complete mathematical exercises demonstrate understanding of the mathematical content involved. Students who have not acquired programming expertise can learn mathematical concepts and skills by working with existing programs. Students can trace programs manually, complete missing lines, replace key lines, add documentation, modify output, find errors, or write related programs before being asked to write a complete program for a given concept. Programming activities to promote and reinforce concept development are provided in Chapter Four.

Problem Solving

Problem solving is not just solving problems! Much of the current work on problem solving is based on George Polya's How To Solve It, Mathematical Discovery and Mathematics and Plausible Reasoning in which he identified strategies for problem solving and argued that problem solving is itself a skill to be learned. The computer can be used as a tool for posing and solving problems in a variety of ways. Examples of using computers in problem solving are given in Chapter Five.

Drill, Practice and Tutorial

Historically, the first application of instructional computing was to provide drill and practice. Since the "early days" of slow, hard-copy terminals, techniques have been refined so that:

1. Drill and practice can be given in animated game-like formats.
2. Programs embody features like immediate feedback, instruction, varied praise and unpraise, and management of students' records.
3. Instructional drill and practice programs can be developed without knowledge of computer programming.

The majority of computer applications in schools today are drill and practice in nature. Tutorial programs go beyond drill and practice to allow the computer to be the sole source of instruction. Tutorial programs present material to be learned, question students on the material, provide appropriate remediation, administer tests, and prescribe what is to be learned next. Drill and practice, and tutorial applications are discussed in Chapter Six.

Management of Instruction and Other Uses of the Computer

Computer Managed Instruction (CMI) is used by teachers who wish to implement individualized instruction. Typically a CMI program has the following components:

1 Classification of the curriculum into objectives and/or strands.

2 A way to assess student progress and generate reports.

3 A way to prescribe what is to be learned based on assessment of progress.

Other applications of computers include: storage and retrieval of information, generation of worksheets and tests, communications networks connecting schools to outside organizations, instruction in utilities, and demonstration applications. Although some of these applications are used most often in media centers or other discipline areas, mathematics teachers should be familiar with them because: professionally, they are often called upon to design or coordinate school-wide computing plans; and personally, some of these applications have important home uses. Chapter Seven presents some of these other applications.

The preceding descriptions give a general overview of the ways that computers are being used in mathematics education. Although most of the applications are broad enough to span the K-12 mathematics curriculum, some are more appropriate than others at particular grade levels. Table I gives some guidance about what is appropriate.

TABLE I
INSTRUCTIONAL COMPUTING BY GRADE LEVEL*

Level	Comments
K - 3	-Students can run computer programs -Enthusiastic users--especially enjoy instructional games -Most are too young to learn formal programming languages
4 - 6	-Students can learn formal programming languages -Simple programming activities to develop concepts can be given -Primary focus still on running instructional programs
7 - 8	-Recommended grades for computer literacy experiences for all students -Increased emphasis on programming to promote the learning of mathematics
9 - 12	-Full range of instructional computing activities in mathematics classes -Computer Science Classes

*Adapted from Apple Computer Company's "Educational Materials Packet"

EPILOGUE

There is little doubt that the use of computers in mathematics education is in its infancy. There is also little doubt that preparations to implement educational computing take more than simply acquiring the hardware, and a brief introduction to the field. As we move through the decade of the 80's, the following are some immediate needs:

1. Funding. Funds to acquire and maintain hardware and software as well as to provide training programs for staff are required. Attention must be given to both the quality and quantity of hardware and software. Support for coordination of computing activities must also be given.

2. Education of Educators. As computers find their way into homes, teachers are finding themselves in the position of knowing less about computers than their students. There is a need for colleges, school districts, and state departments of education to provide computer training for teachers. Educators who have experience and expertise should offer courses or inservice programs for teachers and administrators. Individual teachers can help each other by sharing experiences and resources. Professional organizations can help facilitate these sharing sessions.

3. Instructional Materials. There is no mathematics textbook series that gives more than token attention to computers. In most

cases, the computer is used simply as an optional feature to introduce programming rather than to use programming to develop mathematics.

4. Instructional Support Programs. There certainly is a lack of high- quality instructional support programs. Like textbooks, computer programs vary from excellent to terrible. Mathematics educators and computer programmers should work together to translate good instruction into good computer programs.

5. Coordination. Provisions for coordination of all computing activities within a school or school district must be made. An organized computing library is essential. Individuals with expertise in educational computing should be assigned to coordinate these activities.

6. Communication. Although mathematics and science education journals have begun to attend to issues in educational computing, better communication about equipment (hardware), programs (software), instructional materials and methods, teacher training, successes, and failures is needed. These issues must be addressed within the context of the mathematics curriculum.

7. Public Relations. There is a necessity for a computer literate populace that understands rather than fears technological advances. The public must be made aware of the potentials and limitations of computer applications in education.

8. Research. A great deal of information is needed about learning

and instructional computing. Materials must be developed, field-tested, and revised. Questions related to achievement, attitudes, and processes must be examined. Much of the research that has been completed does not relate to the technology and methodologies of today. Researchers must cooperate with practitioners to maximize the limited resources currently available.

Mathematics education is moving into a new and exciting technological age. The challenges facing educators are great if we are to make optimal use of the emerging technologies. Computers of all "sizes" and types will be needed in schools as students prepare for the technological society in which they will work and live. Teachers also will have to adapt and develop new educational methods which make the computer an integral part of the teaching- learning process. For some, this will require substantial investments of time and energy. Most important, however, is the design, implementation, and continual revision of a coordinated, comprehensive educational computing plan.

Plans for hardware selection, acquisition, maintenance, security, and access must be included. Software required for the chosen applications must be evaluated, selected, and perhaps modified. A procedure for organizing and classifying the software as well as for making it available to students and staff must be devised. Training for school staff must also be considered. The training can take many forms, (e.g., courses, workshops, printed materials, conference attendance), and should foster sharing of experiences and resources . Chapter Eight contains a collection of useful resources.

The computer is only a tool--granted a powerful one--for teaching and learning. As with any tool, computers will only be as effective in the teaching- learning process as teachers and learners are creative in their use.

References

National Council of Supervisors of Mathematics. Position Paper on Basic Mathematical Skills. Arithmetic Teacher, 1977, 25(1), 19-22.

National Council of Teachers of Mathematics. An Agenda for Action: Recommendations for School Mathematics for the 1980s. Reston, VA: NCTM, 1980.

Polya, G. How To Solve It. Princeton, NJ: Princeton University Press, 1957.

Polya, G. Mathematical Discovery. (2 volumes). New York: John Wiley and Sons, 1966.

Polya, G. Mathematics and Plausible Reasoning. (2 volumes). Princeton, NJ: Princeton University Press, 1954.

CHAPTER TWO

COMPUTER LITERACY

Al Monetta
Waterford-Mott High School
Waterford, Michigan

Betty Shaw
Flint Public Schools
Flint, Michigan

Computer literacy has been identified as a basic skill by the National Council of Teachers of Mathematics and the National Council of Supervisors of Mathematics (NCTM, 1977). However, there is little agreement on what computer literacy is, what should be included, or when and how computer literacy should be taught to students.

Because computer hardware, applications, and impact are evolving rapidly, the meanings attached to computer literacy will also continue to evolve. For the purpose of this monograph, computer literacy is defined as a set of skills, concepts, and attitudes related to computers which allows an individual to apply general purpose computers in useful and meaningful ways. Depending on the scope of the set of concepts, skills and attitudes, the following examples illustrate the generality of the definition:

1. A computer literate adult understands the uses and limitations of computers as machines built to serve the needs of people, and is able to vote intelligently on issues related to information accessibility or use.

2. A computer literate student is able to use the computer as a tool in problem solving and as a device to control routine tasks.

3. A computer literate mathematics teacher is able to choose from among the various types of applications presented in

this monograph and apply the most appropriate type in a given teaching/learning situation.

No attempt to recommend an ideal scope and sequence for the set of computer literacy skills, concepts and attitudes will be made here. Rather, exemplary projects, materials, and sources are listed and discussed, sources for further information are noted, and a glossary of terms is provided.

COMPUTER LITERACY PROJECTS

A comprehensive collection of computer literacy objectives was compiled by the Minnesota Educational Computing Consortium (MECC). Funded by the National Science Foundation (NSF), a summary of the work was published in the February, 1980 Mathematics Teacher (pp. 91-96). Objectives were organized into six categories: hardware; programming and algorithms; software and data processing; applications; impact; and attitudes, values, and motivation. Sixty-three objectives were listed which specify skills related to the six categories. See Luehrmann (1981) for an alternative set of objectives which emphasize programming. MECC has also developed a computer literacy questionnaire which is available from MECC. (Its address can be found in Chapter Eight). A final report of the project which describes courses and materials reviewed in organizing the objectives is also available from MECC.

The Human Resources Research Organization (HumRRO), under a grant from the NSF, developed a guide for a K-8 computer literacy

curriculum. Objectives are organized under seven strands by grade level. The strands include: vocabulary, using programs, fundamental concepts, procedures, applications, impact and programming. Programming is included as an optional strand. Objectives and corresponding activities are divided into four grade level groups: K-2, 3-4, 5-6, and 7-8. A series of booklets is available from HumRRO (see Chapter Eight for the address).

The Tri-County Computer Consortium (Wayne, Oakland and Macomb Counties, Michigan), produced a computer literacy curriculum guide, "Computer Literacy and Awareness for Students" (CLAS), which has elementary, junior high, and senior high components. Five general computer literacy goals were specified for students. They are: understanding what computers are and how they evolved, appreciation of the capabilities and limitations of computers, knowledge of careers related to computers, skill development in both using existing programs and writing programs, and awareness of the implications of computers in society. Contact one of the Intermediate School District offices for further details (See Chapter Eight for the address).

Although other computer literacy projects have produced curricular materials, the three projects described above illustrate current approaches to computer literacy. Materials were designed to include hands-on experiences with computers. These experiences were flexible enough to be used with most types of computer equipment. Activities were designed to be integrated into a variety of content areas. It is important to note that computer literacy is an interdisciplinary concern and should be integrated into all content curricula. Presently

however, primary responsibility for computer literacy rests with teachers of mathematics. Thus many activities and objectives were specifically designed for or are easily adaptable to the mathematics classroom.

The goals and objectives in these projects provide a global perspective of what computer literacy means for general populations. Individual teachers and school districts must select and modify these goals and objectives to attend to local curricular goals and hardware facilities. Teachers must become computer literate themselves, and share experiences and information with their colleagues in all subject areas. The general computer literature is rich, but teachers must go beyond the general literature to education- specific sources. This monograph is intended to provide some of this information but other print materials specific to computer literacy in the classroom are listed below. A glossary of terms which will be useful in other chapters of this monograph as well as in the classroom is also provided.

Computer Literacy Resources

Ahl, David, ed. The Best of Creative Computing (volumes 1, 2, 3). Morristown, N.J.: Creative Computing Press, 1976, 1977, 1980.

Ball, M. and Charp, S. Be a Computer Literate. Morristown, N.J.: Creative Computing Press, 1977.

Billings, K. and Moursund, D. Are You Computer Literate. Portland, Oregon: Dilithium Press, 1979.

Collins, Dianne. Computer Awareness in the Elementary School, The

Computing Teacher, Vol. I, No. 1, September, 1979, p. 6.

Horn, C. and Poirot, J. Computer Literacy: Problem Solving with Computers. Austin, Texas: Sterling Swift Co., 1981.

Kurshan, Barbara. Computer Literacy in Practical Ways to Teach the Basic Mathematical Skills, Richmond, VA: Virginia Council of Teachers of Mathematics, 1978.

Lavine, Richard. The Horse Before the Cart: Computer Literacy, The School Administrator, April, 1981, pp. 22-23.

Neill, Michael. Recommendations for Implementing Computer Literacy in Oregon Secondary Schools. Report to the Commission on Computers in Education, September, 1976. University of Oregon, Eugene, Oregon.

Rice, J. My Friend - The Computer. Minneapolis, Minnesota: T. S. Denison and Co., 1976.

Rogers, Jean. An Introduction to Computers and Computing: A High School Course Outline, The Computing Teacher, Vol. 8, No. 7, pp. 30-33.

The Age of Miracle Chips, Time, February 20, 1978, p. 44-59.

Thomas, Rex. Computer Literacy in the High School. AEDS Journal, Spring, 1976, pp. 71-77.

Glossary

ALGORITHM - a step-by-step procedure for completing a task or solving a problem.

ARITHMETIC/LOGIC UNIT - a component of the CPU that carries out arithmetic and logical operations.

ASSEMBLY LANGUAGE - a highly symbolic language which can be translated directly to machine language.

BASIC - Beginners All-Purpose Symbolic Instruction Code - a popular computer language used for most microcomputers.

BIT - Binary Digit - the smallest unit of information, symbolically, either a 0 or a 1.

BUG - an error in a program.

BYTE - a unit of storage, usually made up of eight bits which represents a single keyboard character.

CAI - Computer Assisted Instruction

CHARACTER - a single keyboard symbol, a letter, digit or punctuation mark.

COBOL - Common Business Oriented Language - a computer language most often used in business applications.

COMPUTER - an automatic data processor. A machine capable of operating according to a stored set of instructions.

CONTROL - the unit of the CPU which directs the operation of the other units of the computer.

CPU - Central Processing Unit - the heart of a computer, made up of the arithmetic/logic, and control units.

CRT - Cathode Ray Tube - a computer monitor, or television.

DATA - the symbols processed by a computer. Data can include programs, numbers, or text.

DATA PROCESSING - storing, retrieving and organizing raw data.

DEBUGGING - the process of finding and correcting errors in a computer program.

DISKETTE - a magnetic medium for storing data and programs (floppy disk).

DOS - Disk Operating System - a system program which controls the operation of the disk drive.

FORTRAN - Formula Translation - a language used most often in scientific and mathematical applications.

HARDWARE - the physical parts of a computer system.

INPUT - the process of putting data into a computer. A device used for entering data.

K (KILO) - a prefix representing 1024 (2^10) e.g. a kilobyte KB means 1024 bytes.

KEYBOARD - the most common input device for personal computers. A typewriter-like device for entering data.

LOGO - a high level, graphics oriented language which can be learned by young children. One aspect of LOGO is the creation of graphics by controlling a computer "turtle".

MACHINE LANGUAGE - binary language, a series of 1's and 0's to which all programs must be eventually translated.

MEMORY - the unit of a computer which stores data and programs.

MINICOMPUTER - a medium scale computer usually requiring special facilities and attendants. Often used for time-sharing applications.

MICROCOMPUTER - a stand-alone computer having a CPU on a single integrated circuit. A personal

computer.

MODEM - Modulator-Demodulator - a device used to allow a computer to communicate with another device over the telephone lines.

MONITOR - a CRT, a television used as an output device.

NETWORK - a group of computers or people sharing common communications channels.

OUTPUT - the end result of a computer procedure.

PASCAL - a high level computer language which lends itself to structured programming techniques.

PROGRAM - a sequence of instructions which tell a computer what to do.

RAM - Random Access Memory - User addressable memory -- memory where users' programs are stored. RAM is emptied when the power is turned off.

ROM - Read Only Memory - Memory which is permanent -- is not lost when power is turned off. ROM may not be changed by the user.

SOFTWARE - programs and procedures which make hardware perform required tasks.

STRUCTURED PROGRAMMING - a programming style where a large project is broken up into functionally complete smaller pieces that are individually easier to program, document, and read.

TERMINAL - an input and output device. Allows communication between computer and user.

TIME-SHARING - a process whereby many users take turns using a single CPU. The speed at which the

CPU operates allows the computer to slice a unit of time into thousands of parts and allocate them to individual users.

USER - the person who is controlling the computer.

WORD PROCESSING - text processing, a method of typing, editing and managing text materials.

References

Johnson, D., Anderson, R., Hansen, T., and Klassen, D. Computer Literacy - What Is It? The Mathematics Teacher, 73 (February, 1980), pp. 91-96.

Luehrmann, A. Computer Literacy - What Should It Be? The Mathematics Teacher, 74 (December, 1981), pp. 682-86.

NCTM, National Council of Teachers of Mathematics position paper on basic skills, The Arithmetic Teacher, October, 1977, pp. 19-22.

CHAPTER THREE

TEACHING BASIC PROGRAMMING

Carol Cody
Comstock High School
Comstock, Michigan

Betty Shaw
Flint Public Schools
Flint, Michigan

Mathematics teachers in increasing numbers are being called upon to teach computer programming as well as mathematics. In this chapter, two approaches to teaching computer programming are discussed and key programming concepts are illustrated.

The two approaches are: integrating the teaching of programming into existing courses, and teaching programming as a separate course or sequence of courses. Both approaches require access to computer equipment and teachers with good backgrounds in computer programming. The approaches differ in that the integration approach has the learning of mathematics as the major goal, while the programming course focuses on computer science goals.

Regardless of the approach taken, key programming concepts must be developed. These include: the concept of algorithm, basic logic, distinction between immediate and deferred statements, variables and methods of assigning values, formatting of output, looping and iteration, built-in functions, documentation, and debugging techniques.

Although several computer programming languages are taught in schools, BASIC (Beginners All-Purpose Symbolic Instruction Code) is by far the most commonly studied. BASIC is relatively easy for secondary students to learn and is the primary language used with today's personal computers. Although some upper elementary students have studied BASIC, there are informal ways of introducing programming concepts to young children and some languages, notably LOGO, which young children can use easily. Calculator activities and control of electronic toys(e.g., Bigtrack) can be used to develop looping procedures, and debugging concepts.

The chapter is divided into two parts according to the two approaches to teaching BASIC. The key concepts are addressed in the integration section, and organization, management, and references are presented in the programming course section.

Integrating BASIC into the Mathematics Curriculum

> The use of electronic tools such as calculators and computers should be integrated into the core mathematics curriculum (NCTM, 1980)

Integration of programming within a mathematics course serves to motivate and reinforce the mathematics content as well as extend the range of applications. Examples of developing and reinforcing mathematics concepts with computer programming experiences can be found in Chapter Four. In this section, a plan of action for integrating programming into a course is presented, and BASIC programming skills are illustrated through problems and

generalizations drawn from the problems. Note that many versions of BASIC exist, and each type of computer has its own dialect of BASIC.

An integrated approach to programming in a mathematics course often includes a preliminary unit on using the particular hardware available for the course and introduction of BASIC programming concepts. Teaching the unit at the beginning of the year offers the advantage of maximizing the number of appropriate opportunities for computer enhancement. Some topics lend themselves to computer enhancement readily and others are better done without the computer. If students have experienced the preliminary unit early in a course, the continuity of the course will not be interrupted when students are to write programs for particular topics. There is also the possibility of interference among the programming skills and concepts and mathematics skills and concepts. A preliminary unit early in the year may help minimize this effect.

Initially, students must experience using the hardware i.e., typing on a keyboard, turning machines on and off, executing and interrupting programs, and care and handling of equipment and storage media. Students must have hands-on experience with these procedures. Manufacturer's equipment manuals and other reference materials should also be introduced and available throughout the course.

Students should RUN and LIST some existing programs. The command RUN causes the program in memory to begin executing and is an example of a direct (immediate mode) command. Typing LIST will cause the computer to print the BASIC program code. All lines of the program

begin with a line number. The computer follows the numbers in sequence when executing the program. Any instruction statement which follows a line number is part of a program (deferred mode). The instruction will be carried out only when the program is executed (RUN).

The concept of variable plays an important role in computing. A variable can be thought of as the name of a location in the computer's memory. Numeric, string, and array variables are available in BASIC. NUMERIC VARIABLES can contain only number values; STRING VARIABLES can contain any character values (letters or numbers); and ARRAYS are matrices of characters. Each machine has specific formats for variables; some allow words as variables, some allow multidimensional string arrays, etc., so the examples below use general forms which work on most machines.

The following activities are useful in introducing these various introductory concepts. Students should enter and run the programs on the computer. To facilitate giving all students hands-on experience, have students work in small groups, e.g., 3-5 per group, rotate the groups to the machines and have individual students within the groups take turns typing and running programs.

This program demonstrates some uses of the PRINT statement. Remind students to press the RETURN or ENTER key at the end of each line. Students should type the program, RUN the program and then LIST. Discuss the output.

```
10  PRINT "HELLO"
20  PRINT 6 + 7
30  END

?RUN
HELLO
13
```

Note that anything enclosed in quotes in a PRINT statement will be printed exactly as it appears in the program. Arithmetic operations can also be done in PRINT statements. The correct order of operations is built into the machine. In the following program a value is assigned to the variable G.

```
?LIST

10  PRINT "BEGIN"
20  LET G = 6
30  PRINT G
40  LET G = G + 1
50  PRINT "G NOW HAS THE VALUE";G
60  END

?RUN
BEGIN
6
G NOW HAS THE VALUE 7
```

Note that the equal sign has a special meaning in BASIC. It should be thought of a "takes the value" rather than "equals."

The following program contains a loop. This particular loop is an example of a programmer's nightmare -- an infinite loop. The program interrupt must be used to stop this program. Consult your manual to see how your computer can be told to break out of programs. Some examples are: press BREAK key, press CONTROL and C key simultaneously.

```
10  LET C = 1
20  PRINT C
30  LET C = C + 1
40  GOTO 20
50  END

?RUN
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

BREAK IN 20
```

Some examples of discussion questions are: 1. How would the output change if line 40 were changed to read: 40 GO TO 10? (all 1's printed); 2. Add a comma to the end of line 20 and RUN the program again. What happens? (numbers printed across the page); 3. How could you make the program print out only the odd numbers? (30 LET C=C+2). Many other modifications can be made. Students are fascinated by loops and should be given an opportunity to experiment witn these simple programs. GO TO is the BASIC unconditional branch statement. To make a program stop automatically, a conditional branch statement is used. The IF/THEN statement is the easiest to use. Suppose we want to have the computer count by two's, beginning at 10 and ending at 70. The following programs will each output the required numbers.

```
?LIST
10  LET C = 10
20  PRINT C,
30  IF C = 70 THEN 60
40  LET C = C + 2
50  GOTO 20
60  END

?RUN
10          12          14          16          18
20          22          24          26          28
30          32          34          36          38
40          42          44          46          48
50          52          54          56          58
60          62          64          66          68
70
```

```
?LIST
10  LET C = 10
20  PRINT C,
30  LET C = C + 2
40  IF C = 72 THEN 60
50  GOTO 20
60  END
```

```
?LIST
10  LET C = 10
20  IF C = 72 THEN 60
30  PRINT C,
40  LET C = C + 2
50  GOTO 20
60  END
```

The condition within the IF/THEN statement causes the program to exit the loop when the condition is true. Note that it is important to synchronize the conditional within the IF/THEN statement with the placement of the IF/THEN statement in the sequence of the program. Note also that there is no single program to do any programming task. The fact that a class of 30 students could conceivably write 30 separate programs for the same assignment should encourage teachers to require a RUN of student programs so that the machine does the checking! Once students begin working with loops, the concept of debugging must be discussed.

A program BUG is an error, and DEBUGGING is the process of finding and correcting program errors. Debugging is a form of problem solving and although some strategies are available, success usually depends on individual skills, creativity, and perseverance. Some techniques include: look for typographical errors, print intermediate values, trace through the program by hand, try mini-experiments in separate

and simpler programs, and analyze error messages. "Playing computer" or simulating the RUN of a program should be done early and often in the student's experience. In the program above, a common error produces output ending too soon (at 68) or beyond the goal (72). Examining the placement of the IF/THEN can be done by choosing a simpler case e.g. the numbers 10 to 20, and tracing through the program by hand. As programs are written for complex tasks, more time may actually be spent in the debugging process than any other phase of completing the task. One wag has epitomized debugging by defining a program as "what is left of the path that bugs follow after the bugs have been killed off!"

Solving a problem with the aid of a computer program is not a straightforward process. The problem must be well-defined and solvable by some combination of algorithmic processes. A model must be constructed; a program written, run, and debugged; and the solution examined for reasonableness, and perhaps extended or generalized. Programmers often use flowcharts to help analyze the problem, break the problem into parts, and allow the writer to see that each programming statement directs the program toward its destination. There are standard flowcharting symbols for common functions (start/stop, input/ output, assign/calculate and test conditions). The interested reader can find these in most programming texts listed at the end of this chapter. The following problem and generalizations of it illustrate the BASIC concepts discussed above and introduce other BASIC words and functions.

Consider a form of the classic penny-a-day problem. You earn 1

cent the first day on the job, 2 cents the second day, 4 cents the third day and continue to double your earnings for 30 days. What is the wage on the 30th day? What were the total wages for the month?

A useful strategy for solving the problem is to make a table of values. The columns should include: day, wage for the day, and total earnings. Depending on the mathematical experience of students, the number patterns in the table may be sufficient to generalize solutions. Altering the conditions of the problem in a number of ways quickly demonstrates the efficacy of having an initial computer program to modify.

To write the program, follow the same procedure used in making the table.

TABLE	PROGRAM PROCEDURE	PROGRAM STATEMENT
set up headings enter initial values	choose variable names, D for day, W for wage, T for total earnings - initialize variables	100 LET D=1 110 LET W=1 120 LET T=1
	print values	130 PRINT D,W,T
begin next row (day)	increment	140 LET D=D+1
calculate wage	double last day's wage	150 W=2*W
calculate total	add new wage to old total	160 T=T+W
is table complete? if complete stop, if not do next row	has day 30 been completed? if day is 31 (since the value for day is increased before we test), end, if not, loop back to do next day's calculations	170 IF D=31 THEN 190 180 GO TO 130 190 END

Some computers have only six significant digit accuracy and will give large values in exponential notation. This can lead to discussion of exponential notation and one of the limitations of computers over "paper and pencil" methods. A run of the above program produces the following output. Note that the values are in cents.

```
?RUN
1             1             1
2             2             3
3             4             7
4             8             15
5             16            31
6             32            63
7             64            127
8             128           255
9             256           511
10            512           1023
11            1024          2047
12            2048          4095
13            4096          8191
14            8192          16383
15            16384         32767
16            32768         65535
17            65536         131071
18            131072        262143
19            262144        524287
20            524288        1048575
21            1048576       2097151
22            2097152       4194303
23            4194304       8388607
24            8388608       16777215
25            16777216      33554431
26            33554432      67108863
27            67108864      134217727
28            134217728     268435455
29            268435456     536870911
30            536870912     1.07374182E+09
```

After the program has been developed and RUN (perhaps as a class activity) extensions are in order.

1 What if we consider a 31 day month? (Simply change line 170 to test for 32 rather than 31)

2 What if the wage is tripled each day? (line 150 becomes 150 W=3*W)

3 What if the wage is increased by 10 per cent each day? (150 W = .1*W+W)

To generalize the program so that these values can be set by a user rather than by the programmer, modify the program to be interactive. The INPUT statement is one way to accomplish this.

```
?LIST

10  PRINT "WHAT IS THE ORIGINAL WAGE?"
20  INPUT W
30  PRINT "WHAT PERCENT IS THE WAGE BEING"
40  PRINT "INCREASED?  USE A DECIMAL VALUE."
50  INPUT P
60  PRINT "HOW MANY DAYS?"
70  INPUT E
100  LET D = 1
120  LET T = W
130  PRINT D,W,T
140  LET D = D + 1
150  LET W = P * W + W
160  LET T = T + W
170  IF D = E + 1 THEN 190
180  GOTO 130
190  END

?RUN
WHAT IS THE ORIGINAL WAGE?
?1
WHAT PERCENT IS THE WAGE BEING
INCREASED?  USE A DECIMAL VALUE.
?.5
HOW MANY DAYS?
?10

1             1             1
2             1.5           2.5
3             2.25          4.75
4             3.375         8.125
5             5.0625        13.1875
6             7.59375       20.78125
7             11.390625     32.171875
8             17.0859375    49.2578125
9             25.6289063    74.8867188
10            38.4433594    113.330078
```

To print only the final day's wage and total, place the PRINT statement at the end of the loop. Note that using the INPUT statement is another way to assign values to variables. When the program is RUN, the program stops and waits for the user to enter the requested value. Some computers prompt the user with a question mark.

The program may be modified so that it can be used as a general

purpose compound interest program. The key changes are in line 150 where the day's wage becomes interest for the compounding period. Appropriate user instructions are also required.

```
?LIST

 10  PRINT "WHAT IS THE PRINCIPLE?"
 20  INPUT P
 30  PRINT "WHAT IS THE RATE OF INTEREST?"
 40  PRINT "ENTER A DECIMAL EQUIVALENT."
 50  INPUT R
 60  PRINT "HOW MANY TIMES PER YEAR IS THE"
 65  PRINT "INTEREST TO BE COMPOUNDED?"
 70  INPUT C
 80  PRINT "HOW MANY YEARS?"
 90  INPUT Y
100   LET D = 0
110   LET X = C * Y
120   PRINT "PERIOD", "INTEREST", "PRINCIPLE"
130   PRINT D,I,P
140   LET D = D + 1
150   LET I = P * R / C
160   LET P = P + I
170   IF D = X + 1 THEN 190
180   GOTO 130
190   END

?RUN
WHAT IS THE PRINCIPLE?
?200
WHAT IS THE RATE OF INTEREST?
ENTER A DECIMAL EQUIVALENT.
?.1
HOW MANY TIMES PER YEAR IS THE
INTEREST TO BE COMPOUNDED?
?4
HOW MANY YEARS?
?2
PERIOD          INTEREST              PRINCIPLE
0               0                     200
1               5                     205
2               5.125                 210.125
3               5.253125              215.378125
4               5.38445313            220.762578
5               5.51906445            226.281643
6               5.657041067           231.938684
7               5.79846709            237.737151
8               5.94342877            243.68058
```

Line 110 calculates the total number of compoundings for the

number of years requested. Line 120 prints a heading for the table. The variable D now stands for the current period being calculated rather than the day being calculated. D is initialized at zero so the original principle appears on the first line of output. REM (remark) statements should be used within the program to describe variables and procedures. REM statements are ignored by the computer and serve to document the program for easy reading or revision. An example of an appropriate REM for this program is:

```
95 REM D STANDS FOR THE CURRENT PERIOD
```

The program could be generalized and improved by using FOR/NEXT statements or including standard interest formulas but the nature of compounding and exponential growth is made less abstract by using the repetitive powers of the computer. There are many other statements in the BASIC language and a summary table of a few of them can be found at the end of this section. Another example of developing BASIC concepts by generalizing a program can be found in Chapter Six. Many examples of developing mathematical concepts through programming can be found in Chapter Four.

A BASIC SUMMARY

Keyword	Purpose	Example
LET	Computes and assigns values	10 LET X=3*Y^2
PRINT	Prints numbers, labels, and blank lines	15 PRINT "X= ";X
END	Last statement in a program	87 END
READ	Reads data from a DATA statement	20 READ A,B
DATA	Used with READ to list data	35 DATA 7,-3, .5,8.4
INPUT	Reads data entered from keyboard	16 INPUT A
FOR/TO(STEP)	Sets up and operates a loop (STEP is optional)	25 FOR J=0 TO 10 STEP 2
NEXT	Closes a loop begun with FOR	30 NEXT J
GOTO	Unconditional transfer of control to a specitic line	18 GOTO 10
IF/THEN	Conditional transfer of control to a specific line or statement	40 IF X=5 THEN 30
REM	Permits comments in program	5 REM Program remark

A Computer Course in the Secondary Mathematics Curriculum

> Secondary school computer courses should be designed to provide the necessary background for advanced work in computer science (NCTM, 1980).

The above recommendation supports the growing trend of offering computer science courses in the mathematics department. Many senior high schools currently offer a semester or year course in computer programming. Advanced courses in senior high schools and introductory courses in junior high schools are being offered in increasing numbers.

Algebra I is often a prerequisite for a computer programming course. This prerequisite assures student familiarity with key concepts like "variable", and "formulas", but allows many students the opportunity for learning a programming language like BASIC. Computer science topics other than programming are often treated. These include: history of computers; various number systems like binary, octal and hexadecimal; communication protocols; hardware design; and applications. Since most courses focus on the programming language, it is essential that hardware be available for students during the classroom period and at other times so they may complete assignments.

Courses can be individualized as much as possible by students choosing programs commensurate with their mathematical abilities. Students can design these different programs using the following structured programming techniques:

1. define and analyze the problem, being sure that it merits computer solution
2. clarify the algorithm needed, possibly by constructing a flowchart
3. code the program in BASIC or some other computer language
4. input the program with thorough documentation, and obtain a run with clearly labeled output
5. debug the program, if necessary, and run the program again
6. generalize the program to solve a broader class of problems or "dress up" the program to include better prompts or produce more detailed output.

Students may each complete from six to ten programs and can be evaluated on their assignments, tests, programs, and any special programming projects. A valuable programming project might be to write drill and practice or special purpose programs for use by elementary students.

Such a course needs to be well organized and effectively managed. At the beginning of this individualized course, students can be given the complete programming course requirements for the entire semester. These may include an explanation of programming forms used, followed by the programming techniques learned which are paralleled with the BASIC language requirements, and the respective due dates for each program.

At least six or more computer stations must be available for one class of twenty students. Audio-visual materials may include transparencies, slides and filmstrips. A computer library in the classroom may include various manuals, BASIC texts, problem books, computer application books, periodicals, and a dictionary of computer terms. Field trips to computer centers, guest speakers, and programming contests with other schools can highlight some of the activities.

An outline and teacher's guide of a seven week course in computers and BASIC can be found in Dorn, 1979, pp. 25-54. A classroom configuration scheme and course outline for a semester computer science course can be found in Magel & Hamblen, 1979, pp. 119-126.

The following BASIC textbooks have been used in various programming courses. Each computer has manuals which are specific to its dialect of BASIC. The books listed here are not specitic to any one machine.

BASIC Textbooks

Coan, James. BASIC BASIC. Rochelle Park, N.J.: Hayden Co., 1970.

Dwyer, T. & Crithfield, M. BASIC and the Personal Computer. Reading, MA: Addison-Wesley Co., 1978.

Golden, Neal. Computer Programming in the BASIC Language. (2nd ed.) N.Y.: Harcourt Brace Jovanovich, 1981.

Larsen, Sally. Computers for Kids. Morristown, N.J.: Creative Computing Press, 1981. (Note: separate editions for Apple, Pet and Radio Shack computers)

Lien, David. The BASIC Handbook: An Encyclopedia of the BASIC Computer Language. San Diego, CA: Compusoft, 1978.

Peckham, Herbert. BASIC: A Hands-on Method. N.Y.: McGraw-Hill, 1978.

Spencer, Donald. Using BASIC in the Classroom. Ormond Beach, Florida: Camelot, 1978.

References

Dorn, Christine. Microcomputers and the 3 r's. Rochelle Park, N.J.: Hayden, 1979.

Magel, K & Hamblen, J. Microcomputer Networks for Introductory Computer Science. AEDS JOURNAL, Vol. 13, Fall 1979, pp. 119-126.

NCTM. An Agenda for Action -- Recommendations for School Mathematics of the 1980's. Reston, VA: NCTM, 1980.

CHAPTER FOUR

PROGRAMMING ACTIVITIES FOR DEVELOPING CONCEPTS

Rheta Rubenstein
Renaissance High School
Detroit, Michigan

Students' developing concepts, abstracting, and generalizing are important goals of mathematics instruction. This chapter intends to illustrate how computer programming and programming-related activities can be designed and used in mathematics courses to promote student attainment of these goals. The curriculum envisioned demands minimal programming skills on the part of students. The focus is on prepared programs that students manipulate in a variety of ways to extend traditional practice to higher cognitive levels.

The experience of programming a computer has often been acclaimed as a means for enhancing students' learning of mathematical concepts. Just as in a tutoring situation, the tutor (programmer) must have a complete understanding of the process being taught in order to explain it to a student (computer). The benefits of such an activity for the tutor (programmer) in clarifying and reinforcing one's own understanding are well known.

Writing computer programs, however, is a difficult, multi-step, time-consuming process. There are questions whether certain aspects of

programming such as format manipulation and idiosyncratic details of certain machines ought to be dwelt upon during mathematics instructional time. Indeed, the mathematics teacher would prefer to focus time and effort on mathematical concepts. There are, however, several activities requiring minimal programming background which can be included in a traditional curriculum to extend practice exercises and to enhance concept development. Such programming activities include:

1. completing a program by inserting key formulas
2. annotating a program by inserting remarks or print statements
3. modifying an existing program to perform a related or more general task
4. predicting the output of a given program
5. debugging a nearly correct program
6. selecting data and checking the output of a given program
7. making an inference from an error message
8. comparing two related programs regarding their output or procedures

Of course, the activity, "write and run a program to . . ." is always another option. Several resources provide examples of programs to be written in conjunction with algebra, geometry, or advanced topics. (Coan, 1970; Dwyer and Critchfield, 1977; Golden, 1981) The following materials, in contrast, illustrate what can be done at pre-algebra levels, as well as beyond, without requiring students to construct entire programs from scratch.

It is expected that materials like the following will be stored on tape or disk and will be available for students to load, manipulate (e.g., modify, complete), and run. They may be used by students individually, in small groups, or as teacher-directed class activities.

Once the spirit of these strategies is illustrated and understood, it is hoped that mathematics teachers will begin to recognize opportunities to create similar materials to enhance their own instruction.

Topic: Reading Programs

Level: 4 - 8

Purpose: Students practice interpreting computer programs.

Activities:

Each of the following programs performs a mathematical calculation. Study each program to decide what it does, then write REM and PRINT statements and insert them in the program to make the program and output clearer. Finally, run the programs. Note that programs 2 - 4 produce an out of data error message. Discuss the computer's attempt to provide feedback on programs and how error messages can be used in the important process of debugging.

Program 1.

```
?LIST

10  INPUT A,B,C,D,E
20  LET Y = (A+B+C+D+E) / 5
30  PRINT Y
40  END
```

```
?RUN
?1,2,3,4,5
3
```

Program 2.

```
?LIST

10  READ X
20  LET Y = X ^ 2
30  PRINT Y
40  GOTO 10
50  DATA 1,2,3,4,5,6,7,8,9,10
60  END
```

```
?RUN
1
4
9
16
25
36
49.0000001
64
81.0000001
100

?OUT OF DATA ERROR IN 10
```

Program 3.

```
?LIST

10  LET S = 0
20  READ N
30  LET T = N ^ 2
40  LET S = S + T
50  PRINT N,T,S
55  GOTO 20
60  DATA 1,3,5,7,9,11,
        13,15,17,19
70  END
```

```
?RUN
1           1           1
3           9           10
5           25          35
7           49.0000001  84.0000001
9           81.0000001  165
11          121         286
13          169         455.000001
15          225         680.000001
17          289         969.000001
19          361         1330

?OUT OF DATA ERROR IN 20
```

Program 4.

```
?LIST

10  READ L,W
20  LET P = 2 * L + 2 * W
30  LET A = L * W
40  PRINT P,A
50  GOTO 10
60  DATA 2,5,7,9,10.3,6
70  END
```

```
?RUN
14              10
32              63
32.6            61.8

?OUT OF DATA ERROR IN 10
```

Program 5.

```
?LIST

10  REM  CONVERTS HEIGHTS
20  READ F,I
30  LET H = F * 12 + I
40  PRINT F,I,H
45  IF F = 4 THEN 60
47  GOTO 20
50  DATA 5,2,3,11,4,6,4,10
60  END
```

```
?RUN
5           2           62
3           11          47
4           6           54
```

Topic: Multiplication Arrays

Level: 4- 8

Purpose: Students will be able to associate multiplication facts and arrays and explore the concepts of prime and composite.

Activities: Run the following program. Use 4 and 6 for input.

```
?LIST

10  REM MULTIPLICATION ARRAYS
15  PRINT "HOW MANY ROWS";
20  INPUT A
30  PRINT "HOW MANY STARS IN A ROW ";
40  INPUT B
50  FOR J = 1 TO A
60  FOR I = 1 TO B
70  PRINT "*";
80  NEXT I
90  PRINT
100  NEXT J
110  END

?RUN

HOW MANY ROWS  ?4
HOW MANY STARS IN A ROW ?6
******
******
******
******
```

1. How many stars were printed?

2. Predict the total number of stars that will be printed for each of the following inputs. Run the program to check your predictions.

Rows	2	3	5	5	6	8	9	7	8
Stars in each row	5	4	5	6	7	7	8	9	8
Total stars									

3. Select two numbers of each result below. Run the program to check your predictions.

Rows					
Stars in each row					
Total stars	21	25	27	35	49

4. How is the total number of stars related to the two input numbers?

5. Use the program to print the following number of stars in two different ways. Example: 12 stars

```
******    or    ****
******          ****
2,6             ****
                3,4
```

Write the two pairs for each of the following:

a. 18 stars b. 24 stars c. 28 stars d. 30 stars

*e. Find a number of stars under 50 that can be drawn as three different arrays.

6. List all the numbers of stars under 50 that can be drawn in arrays of at least two rows and two stars per row.

7. List all the numbers of stars under 50 that cannot be drawn as an array of at least two stars per row and at least two rows.

8. Use your textbook or a dictionary to find the mathematical meanings of the words prime and composite. Describe your results to problems 6 and 7 using these words.

*9. List all the numbers of stars under 100 that can be printed in more than one array. Show the different arrays for each.

*10. If you have studied nested loops in programming, change the program above to print an array for each multiplication fact from 5 x 5 to 10 x 10.

Topic: Rounding to Nearest Whole Number

Level: 5 - 8

Purpose: Students discover how to program a rounding function and practice rounding to the nearest whole number.

Activities: Run the following program.

```
?LIST

5  REM  MYSTERY FUNCTION
10  READ X
20  DEF  FN R(X) = INT (X + .5)
30  PRINT  FN R(X)
35  IF X = 99.5 THEN 50
37  GOTO 10
40  DATA 2.9,5.45,6.08,97.08,15.1,1.51,1.009,99.99,99.5
50  END

?RUN
3
5
6
97
15
2
1
100
100
```

1. Describe what the program does. If you need, rewrite line 40 to include your own data.

2. Predict the output for the following input. Test you predictions by running the program.

DATA: 3.7 1.4 1.6 62.5 12.09 18.38 .03

.07 19.6

OUTPUT:

3. Modify the program so that it will round input numbers to the nearest tenth.

Topic: Decimal Equivalents of Rational Numbers

Level: 6 - 9

Purpose: Students identify repeating and terminating decimal equivalents of rational numbers and explore conditions which account for the repeating or terminating result.

Activities: The following program will print the first several decimal places in the decimal numeral equivalent to a given rational number in fractional form.

```
?LIST
5  REM  DECIMAL EQUIVALENT OF A GIVEN FRACTION
10   PRINT "INPUT NUMERATOR N ";
20   INPUT N
30   PRINT "INPUT DENOMINATOR D ";
40   INPUT D
50   PRINT N"/"D"=.";
60   LET K = 0
70   LET Q = 10 * N / D
80   LET M = INT (Q)
90   PRINT M;
100   LET N = 10 * N - M * D
110   IF N = 0 THEN 150
120   LET K = K + 1
130   IF K > 2 * D THEN 150
140   GOTO 70
150   END

?RUN
INPUT NUMERATOR N  ?3
INPUT DENOMINATOR D  ?4
3/4=.75
]RUN
INPUT NUMERATOR N  ?4
INPUT DENOMINATOR D  ?11
4/11=.363636363636363636363
]RUN
INPUT NUMERATOR N  ?7
INPUT DENOMINATOR D  ?8
7/8=.875
]RUN
INPUT NUMERATOR  N ?2
INPUT DENOMINATOR D  ?3
2/3=.6666666
]RUN
INPUT NUMERATOR N  ?4
INPUT DENOMINATOR D  ?7
```

```
4/7=.571428571428571
]RUN
INPUT NUMERATOR  N  ?2
INPUT DENOMINATOR D  ?5
2/5=.4
```

1. Run the above program to find decimal equivalents of 3/4, 4/11, 7/8, 2/3, 4/7, 2/5.
2. A decimal numeral which stops or ends in zeros is called a terminating decimal. For example 3/4 = .75. A decimal numeral which repeats without terminating is called a non-terminating, repeating decimal. For example, 4/11 = .36363636. . . The two digits 36 would repeat forever if the computer were not programmed to stop.

   Use the above program to decide whether the following fractions have terminating or non-terminating repeating decimal equivalents: 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9, 1/10.
3. Find three more fractions with terminating decimals. Record your results.
4. Find three more fractions with non-terminating, repeating decimals. Record your results.
5. Make a table of all the fractions you found with terminating decimals. One example is given.

TERMINATING DECIMALS

N	D	N/D	Number of digits in decimal
3	4	.75	2

6. Make a table of all the fractions you found with non-terminating repeating decimals. Two examples are given.

NON-TERMINATING REPEATING DECIMALS

N	D	N/D	Number of repeating digits
4	11	.36363636...	2
2	3	.6666667 ...	1

7. Study the denominators in the two tables above. How are the denominators related to whether the decimal terminates or repeats? (Hint: Make another column giving the prime factorization of each denominator.)

8. Predict whether the following fractions will repeat or terminate. Check your prediction with the computer.
1/16, 1/18, 1/20, 1/24, 1/25, 1/26, 1/30, 1/32, 1/40.

*9. Try to discover a pattern in your table of repeating decimals. Can you predict the number of repeating digits?

Topic: Distributive Property

Level: Algebra

Purpose: Students explore the distributive property of multiplication over addition and other possible distributive properties.

Activities:

1. The Distributive Property of Multiplication over Addition states that: A * (B + C) = A*B + A*C

   For example: 3 * (4 + 5) = 3*4 + 3*5

   because 3 * 9 = 12 + 15.

   Choose your own values for A, B, and C and run the following program several times. Can you find any cases when the Distributive Property of Multiplication over Addition is false?

```
?LIST

5  REM DISTRIBUTIVE PROPERTY
10  PRINT "INPUT A ";
20  INPUT A
30  PRINT "INPUT B ";
40  INPUT B
50  PRINT "INPUT C ";
60  INPUT C
70  PRINT A"*("B"+"C") = "A"*"B + C" = "A * (B + C)
75  PRINT
80  PRINT A"*"B"+"A"*"C" = "A * B"+"A * C" = "A * B + A * C

?RUN
INPUT A  ?3
INPUT B  ?4
INPUT C  ?5
3*(4+5) = 3*9 = 27

3*4+3*5 = 12+15 = 27
```

2. Let's see what happens it we change the distributive property above by replacing addition with subtraction. This gives the "Distributive Property of Multiplication over Subtraction. (DPMS)."

   A * (B - C) = A*B - A*C

a. Predict whether there are false cases of the DPMS?

b. Test your conjecture by changing lines 70 and 80 in the program. Replace all the plus signs (+) with minus signs (-). Use trial data to decide whether the DPMS is ever false. State your findings.

3. Consider the following candidates for distributive properties:

a. Distributive Property of Division over Addition

$A \div (B + C) = A \div B + A \div C$

b. Distributive Property of Division over Subtraction

$A \div (B - C) = A \div B - A \div C$

aa. Which of the above are always true?

bb. Rewrite lines 70 and 80 to test each of these "distributive properties." State your results.

4. In all the above "distributive properties" the single number A appeared to the left of the expression in parentheses. These are all examples of what are called "left distributive properties." Below are candidates for right distributive properties:

a. Right Distributive Property of Multiplication over Addition $(B + C) * A = B*A + C*A$

b. Right Distributive Property of Multiplication over Subtraction $(B - C) * A = B*A - C*A$

c. Right Distributive Property of Division over Addition

$(B + C) \div A = B \div A + C \div A$

d. Right Distributive Property of Division over Subtraction $(B - C) \div A = B \div A - C \div A$

Predict which of the above "distributive properties" are

always true. Rewrite lines 70 and 80 to test each of these "distributive properties." State your results.

Topic: Linear Equations and Their Graphs

Level: Algebra

Purpose: Students write BASIC formulas for the slope, intercepts, and slope-intercept form from linear equation given in standard form. They also identify horizontal and vertical lines and points that are on a given line.

Activities:

1. The standard equation for a line in the plan is A*X + B*Y = C. Complete the following computer program so that given A, B, and C (not both A and B equal to zero), it will print:
   a. the slope of the line (line 100)
   b. the y intercept (line 110)
   c. the x intercept, if there is one (line 130)
   d. the equation of the line in slope-intercept form (line 150)
   e. appropriate messages for horizontal or vertical lines (lines 170, 190).

   Test your program with problems which you already solved.

```
?LIST

5  REM LINEAR EQUATIONS
10  PRINT "INPUT REAL NUMBERS A,B,C FOR"
20  PRINT "THE EQUATION AX+BY = C."
30  PRINT "A= ";
40  PRINT A
50  PRINT "B= ";
60  PRINT B
70  PRINT "C= ":
80  PRINT C
90   IF B = 0 THEN 170
100  PRINT "SLOPE= "; - A / B
110  PRINT "ORDINATE OF Y-INTERCEPT IS  "; C / B
120  IF A = 0 THEN 190
130  PRINT "ASCISSA OF X-INTERCEPT IS  "; C / A
140  PRINT "IN Y FORM, "A"X+"B"Y = "C" IS:"
```

```
150  PRINT "Y = ": - A / B; "X+"; C / B
160  GOTO 200
170  PRINT "VERTICAL LINE. X= "; C / A
180  GOTO 200
190  PRINT "HORIZONTAL LINE.  Y= "; C / B
200  END

?RUN
INPUT REAL NUMBERS A,B,C FOR
THE EQUATION AX+BY = C.
A= ?3
B= ?2
C= ?8
SLOPE= -1.5
ORDINATE OF Y-INTERCEPT IS 4
ASCISSA OF X-INTERCEPT IS 2.66667
IN Y FORM, 3X+2Y = 8 IS:
Y = -1.5X+4
```

NOTE to teacher:

Statements underlined

are to be omitted from student

materials.

2. Statements 200-270 will tell whether a given point (R,S) is on the line AX + BY = C. Complete line 200, then test your program with problems you have already solved.

```
?LIST

5  REM LINEAR EQUATIONS
10  PRINT "INPUT REAL NUMBERS A,B,C, FOR"
20  PRINT "THE EQUATION AX+BY = C."
30  PRINT "A= ";
40  INPUT A
50  PRINT "B= ";
60  INPUT B
70  PRINT "C= ";
80  INPUT C
90  IF B = 0 THEN 170
100  PRINT "SLOPE- "; - A / B
110  PRINT "ORDINATE OF Y-INTERCEPT IS ";C / B
120  IF A - 0 THEN 190
130  PRINT "ASCISSA OF X-INTERCEPT IS ";C / A
140  PRINT "IN Y FORM, "A"X+"B"Y = "C" IS:"
150  PRINT "Y = "; - A / B;"X+";C / B
160  GOTO 200
170  PRINT "VERTICAL LINE. X = "; C / A
180  GOTO 200
190  PRINT "HORIZONTAL LINE. Y = ";C / B
200  PRINT "INPUT COORDINATES OF A POINT (R,S)."
210  INPUT R,S
220  IF A * R + B * S = C THEN 250
230  PRINT "("R","S") IS NOT ON THE LINE."
240  GOTO 260
250  PRINT ("R","S") IS ON THE LINE."
260  PRINT A"X+""B"Y = "C"."
270  END
```

```
?RUN
INPUT REAL NUMBERS A,B,C FOR
THE EQUATION AX+BY = C.
A= ?3
B= ?2
C= ?8
SLOPE= -1.5
ORDINATE OF Y-INTERCEPT IS 4
ABCISSA OF X-INTERCEPT IS 2.6666667
IN Y FORM, 3X+2Y = 8 IS:
Y = -1.5X+4
INPUT COORDINATES OF A POINT (R,S).
?1,2
(1,2) IS NOT ON THE LINE
3X+2Y = 8.
```

Topic: Dynamic Equation Solving

Level: Algebra

Purpose: Students use a program to search intervals to locate roots of equations.

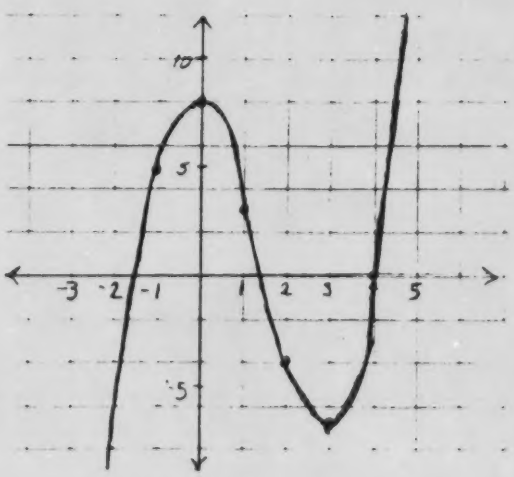
Background: A root of a polynomial equation is a value of x such that $f(x) = 0$. A third degree polynomial such as $f(x) = x^3 - 4x^2 - 2x + 8$ may have 1, 2, or 3 real roots. The following program generates ordered pairs which can be used to get a picture of the function's behavior.

```
?LIST

5  REM  ROOTS OF POLYNOMIAL
10  PRINT "X", "F(X)"
20  DEF FN P(X) = X ^ 3 - 4 * X ^ 2 - 2 * X + 8
30  FOR X = -3 TO 5
40  PRINT X, FN P(X)
50  NEXT X
60  END

?RUN
X                F(X)
-3               -49
-2               -12
-1               5
0                8
1                3
2                -4
3                -6.9999999
4                0
5                23
```

In the intervals where the value of polynomial changes signs the graph must be crossing the x axis. These are the intervals where the zeros are located. A sketch of the graph would look something like this:



Activities:

1. Modify and run the above program to locate the intervals of roots and sketch graphs for each of the following polynomial equations.

   a. $x^3 - 2 = 0$ b. $x^4 - x^3 + 3 = 0$

   c. $x^5 - 2x^4 + 2 = 0$

2. Let's see if we can get a more exact value for these roots. Modify line 30 to help:

   30 FOR X = -2 TO -1 STEP .1

   Use this type of modification to find the roots of each of the equations in #1 to the nearest tenth.

3. Describe how you could find roots to the nearest hundredth Try it for one equation above.

4. The method you have been using requires a lot of work to constantly rewrite your program. On the next page is a program which allows you to input your function, the endpoints of the interval you want searched, and the step size. It then prints your table of values. Also, you may use any function, not just polynomials.

   Use the following program to find real roots of

   a. $f(x) = x^2 - 7$

   b. $f(x) = 2x^3 - 4x^2 + 8$

   c. $f(x) = x^4 - 2.5x^3 + .97x^2 + .5x - .3$

   *d. $f(x) = e^x - x^2$, Note: e^x = EXP(X)

```
?LIST

5  DIM A$(3),B$(3),C(3)
10  DEF FN F(X) = X ^ 3 - 7 * X + 6
20  PRINT "HAVE YOU DEFINED YOUR FUNCTION"
22  PRINT "IN LINE 10?  TYPE YES OR NO ";
```

```
25  INPUT A$:  IF A$ = "YES" THEN 240
30  PRINT "DO YOU WANT INSTRUCTIONS?  YES OR NO"
40  INPUT A$:  IF A$ = "NO" THEN 150
50  PRINT "THIS PROGRAM WILL HELP YOU ISOLATE"
60  PRINT "THE REAL ROOTS OF AN EQUATION.  THE"
70  PRINT  "PROCESS IS BASED ON DESCARTES RULE OF"
80  PRINT "SIGNS.  TO USE THIS PROGRAM YOU MUST
90  PRINT "DEFINE A FUNCTION IN LINE 10 OF THIS"
100  PRINT "PROGRAM, IN THIS FORM:"
110  PRINT TAB ( 15)"10 DEF FNF(X)="
120  PRINT "AFTER THAT THE PROGRAM DOES THE"
130  PRINT "REST BY ASKING YOU TO INPUT THE INTERVAL"
140  PRINT "YOU WANT SEARCHED AND THE INCREMENT."
150  PRINT
160  PRINT "HAVE YOU DEFINED YOUR FUNCTION?"
170  PRINT "YES OR NO.  IF YOU ANSWER NO THE"
180  PRINT "PROGRAM WILL STOP.  TYPE YOUR"
190  PRINT "FUNCTION IN AS LINE 10 AND THEN"
195  PRINT "RUN THE PROGRAM AGAIN."
200  PRINT
210  INPUT B$
220  IF B$ = "YES" THEN 240
230  STOP
240  PRINT
250  PRINT "INPUT THE INTERVAL (A TO B) YOU WANT"
260  PRINT "SEARCHED IN THE FORM A,B."
270  INPUT A,B
280  PRINT
290  PRINT "INPUT THE STEP."
300  INPUT S
310  PRINT
320  PRINT "J=","FNF(J)="
330  FOR J = A TO B + .00005 STEP S
340  PRINT J, INT(FN F(J)*10000 + .5) / 10000
350  NEXT J
360  PRINT "DO YOU WANT TO SEARCH ANOTHER INTERVAL?"
370  PRINT "YES OR NO."
380  INPUT C$
390  IF C$ = "YES" THEN 240
400  END

?RUN
HAVE YOU DEFINED YOUR FUNCTION
IF LINE 10?  TYPE YES OR NO ?YES

INPUT THE INTERVAL (A TO B) YOU WANT
SEARCHED IN THE FORM A,B.
?-5,5

INPUT THE STEP.
?1
```

```
J=                FNF(J)=
-5                -84
-4                -30
-3                0
-2                12
-1                12
0                 6
1                 0
2                 0
3                 12
4                 42
5                 96
DO YOU WANT TO SEARCH ANOTHER INTERVAL?
YES OR NO.
?NO
```

Topic: Sequences and Series

Level: Advanced Mathematics

Purpose: Students will be able to identify a sequence, distinguish arithmetic and geometric sequences, find sums of arithmetic sequences, and study the convergence of certain series.

Activities:

1. Study the following programs. Decide for each one whether the output will be a sequence. If not, why not? If so, what is the domain and what rule can you give for determining the Nth term of the sequence?

```
?LIST

10  FOR N = 1 TO 12
20  LET A = ABS (N-6)
30  PRINT N,A
40  NEXT N
50  END

?RUN
1                5
2                4
3                3
4                2
5                1
6                0
7                1
8                2
9                3
10               4
11               5
12               6
```

```
?LIST

10  FOR N = 1 TO 20 STEP 3
20  LET A = SQR (N ^ 2 - 1)
30  PRINT N,A
40  NEXT N
50  END

?RUN
1                0
4                3.87298335
7                6.92820323
10               9.94987438
13               12.9614814
16               15.9687194
19               18.973666
```

```
?LIST

10  LET N = 1
20  LET A = 4*N^2-3*N+7
30  PRINT N,A
40  IF N = 10 THEN 70
50  N = N + 1
60  GOTO 20
70  END

?RUN
1                8
```

```
?LIST

10  LET P = 1
20  FOR I = 1 TO 10
30  LET P = P * I
40  PRINT I,P
50  NEXT I
60  END

?RUN
1                1
```

```
2              17             2              2
3              34             3              6
4              59             4              24
5              92             5              120
6              133            6              720
7              182            7              5040
8              239            8              40320
9              304            9              362880
10             377            10             3628800
```

2. Study these two programs. Note the distinction between them. Write appropriate PRINT statements to introduce each one. Run the programs.

```
?LIST

10  LET A = 3
20  LET X = 2
30  FOR N = 1 TO 10
40  PRINT A
50  LET A = A * X
60  NEXT N
70  END

?RUN
3
6
12
24
48
96
192
384
768
1536
```

```
?LIST

10  LET A = 3
20  LET X = 2
30  FOR N = 1 TO 10
40  PRINT A
50  LET A = A * X
60  NEXT N
70  END

?RUN
3
5
7
9
11
13
15
17
19
21
```

3. The following program is designed to print the terms of a geometric sequence. Run the program.

```
?LIST

5  REM A GEOMETRIC SEQUENCE
10  FOR N = 1 TO 10
20  LET A = 4 * ( - 2) ^ (N - 1)
30  PRINT A
40  NEXT N
50  END

?RUN

4
-8
16
```

-32
64
-128
256
-512
1024
-2048

a. What is a_1?

b. What is the constant ratio, r?

c. Modify the program to print the geometric sequence in which $a_1 = 1/5$ and $r = 1/2$. Run the new program.

d. Modify the program so the user can input a_1, r, and the number of terms to be printed. Run your new program for several sequences.

4. The sum of an arithmetic sequence $s_n = a_1 + a_2 + \cdots + an$, can be found by the formula $s_n = \frac{n}{2}(a_1 + a_n)$. The computer can find the sum of a sequence by continuing to add the new terms. Write two computer programs to find S_{20} for each of the following sequences. One program should use the formula; the other should not.

a. $a_1 = 12$, $d = 0.25$ b. $a_1 = 12$, $d = -0.25$

5. Write the run programs to examine the behavior of the following series. Describe what you observe.

a. $1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + \ldots$

b. $pi = 4(1 - (1/3) + (1/5) - (1/7) + (1/9) - (1/11) + \ldots)$

c. $e = 1/0! + 1/1! + 1/2! + 1/3! + 1/4! + 1/5! + \ldots$

*6. Find other formulae to generate pi or e. Write the programs based on these series. Compare the efficiency of different formulae.

References

Braunfeld, P. Some Thought on Algorithms and Computers. In physical Science Group, The Cape Ann Conference on Junior High School Mathematics. Physical Science Group: Newton, MA, 1973.

Coan, J. Basic BASIC: An Introduction to Computer Programming in BASIC Language. New York: Hayden Book Co., 1970.

Dwyer, T. and Critchfield, M. Computer Resource Book - Algebra. Boston: Houghton Mifflin Co., 1977.

Fehr, H., Fey, J., Hill, T., and Camp, J. Unified Mathematics Course IV. Menlo Park, CA: Addison-Wesley, 1974.

Golden, N. Computer Programming in the BASIC Language (2nd ed.). New York: Harcourt, Brace, Javonovich, 1981.

Hirsch, C. Computers in Pre-High School Mathematics. Manitoba Association of Mathematics Teachers Newsletter. June, 1973.

Landry, M. Algebra and the Computer. The Mathematics Teacher, 1980, 73 pp. 663-67.

Rogowski, S. Problems for Computer Solution. Morristown, NY: Creative Computing Press, 1980.

CHAPTER FIVE

COMPUTER METHODS FOR PROBLEM SOLVING

Dwayne Channell
Western Michigan University
Kalamazoo, Michigan

Carol Cody
Comstock High School
Comstock, Michigan

Christian Hirsch
Western Michigan University
Kalamazoo, Michigan

In its Agenda for Action the National Council of Teachers of Mathematics recommends that problem solving be the focus of school mathematics in the 1980s. This chapter illustrates how computers may be used to assist pupils in solving problems while simultaneously reinforcing their understanding of processes for solving problems. Ten different problem situations are presented. For each situation a 4-stage problem solving model a la Polya is given: Problem Statement, Analysis, Computer Program, and Looking Back.

The chapter is divided into two sections: Problem Solving Using Algorithmic Methods and Problem Solving Using Simulation Methods. The first section treats a variety of problems whose solutions result from straightforward numerical computation. The solution process involves specifying a computer algorithm for carrying out computations which would be tedious, if not impossible to perform "by hand" or even with the aid of a calculator. In some cases, the method involves modifying a known formula such as that for the volume of a rectangular solid. In other cases, an algorithm specific to the situation needs to be constructed. The results of computing may provide the solution to the

problem or simply tabular data to be analyzed by the student for a possible solution. Computer methods of this sort provided the solution to the famous four-color problem in 1976. The proof required 1200 hours of computer time in addition to a substantial body of mathematical theory.

The second section is devoted to problem situations for which there is no known mathematical algorithm for solution or for which the algorithm is so complicated that requiring its use would make the solution inaccessible to all but advanced high school students. Rather than ignoring such problems, students can be guided to simulate the actual physical situation. That is, they can formulate a mathematical model that may be used to represent or imitate the actual physical phenomena. Once the model has been constructed, a computer program may be developed or supplied to investigate the model over time or under varying conditions. Computer simulations of this type are used in government and industry to investigate the effects of pollutants on our water resources, to design automobiles, and to develop economic policies. Problems involving randomness, such as probability or expected values, can be solved by special simulation techniques known as Monte Carlo methods. Simulations of this type use random numbers to model war games, to predict long range weather patterns, to analyze traffic flow, and to develop statistical tests.

Most of the problems in this chapter can be adapted for use in elementary, junior high, and high school with students of different levels of computer programming ability. The first problem requires no programming skills on the part of the student. The computer program is

to be used as the source of the problems! For the remaining problems computer programs can be written by students themselves or the sample programs supplied to them after the analysis of the problem has been completed. In either case experiences with the roles computers can play in solving problems should increase student understanding of computer methods as well as problem solving processes.

Problem Solving Using Algorithmic Methods

Topic: Recognizing Number Patterns

Problem: One of the fundamental methods of problem solving in mathematics is that of recognizing patterns and making predictions based upon those patterns. For example, given that the first three numbers of a sequence are 1, 2, and 4, can you predict the next number in the sequence? If you said 8, you are correct. The pattern is "multiply the last term by 2 to get the next term." The sequence is 1, 2, 4, 8, 16, 32, 64, and so on.

As a second example, assume the first three numbers of a sequence are again 1, 2, and 4. What is the next number in the pattern? Eight? No, the next number in the sequence I'm thinking of is 7. Now, knowing that the first four terms of the sequence are 1, 2, 4, and 7, can you guess the next number? If you guessed 11, you are correct. The pattern is "begin with 1, add 1 to get 2, add 2 to get 4, add 3 to get 7, add 4 to get 11, add 5 to get 16, etc." The sequence is given by 1, 2, 4, 7, 11, 16, 22, and so on.

Program: The BASIC program given here can be used by teachers at both elementary and secondary levels of instruction. It provides practice

in recognizing number patterns and making predictions based upon the observed patterns. As written, the program presents the user with the three different number patterns provided in the DATA statements at the end of the program listing. The teacher can change the number patterns by changing these DATA statements. If it is desirable to present the student with more (or less) than 3 patterns, this can be done by modifying the DATA statements and by changing the value of N in line 170. The program as written works with the first 10 terms of a sequence. The sequence can be made longer by making the appropriate changes in lines 160, 200, and 400.

```
10  REM PROGRAM PRESENTS USER WITH N
20  REM SEQUENCES ONE AT A TIME.  THE
30  REM VALUE OF N IS DEFINED IN LINE 170.
40  REM EACH TEN TERM NUMBER SEQUENCE IS
50  REM DEFINED WITH A SEPARATE DATA
60  REM STATEMENT BEGINNING WITH LINE
70  REM NUMBER 1000.  PRESENTATION OF A
80  REM SEQUENCE IS ENDED IF 3 CORRECT
90  REM TERMS ARE IDENTIFIED IN SUCCESSION
100  REM  OR IF ALL TEN TERMS OF THE
110  REM  SEQUENCE HAVE BEEN PRESENTED.
115  PRINT : PRINT
120  PRINT "I'M GOING TO SHOW YOU A SET OF NUMBERS."
122  PRINT "I WANT YOU TO LOOK FOR A PATTERN AND"
124  PRINT "THEN TRY TO GUESS THE NEXT NUMBER;
126  PRINT  "THAT FITS THE PATTERN."
130  PRINT : PRINT : PRINT : PRINT
160  DIM A(10)
170  LET N = 3
180  LET S = 1
190  LET C = 0
200  FOR I = 1 TO 10
210  READ A(I)
220  NEXT I
230  LET T = 3
240  PRINT "THE FIRST ";T;" NUMBERS IN THE PATTERN ARE:"
245  PRINT
250  FOR I = 1 TO T - 1
260  PRINT A(I);",";
270  NEXT I
280  PRINT A(T)
290  PRINT : PRINT
300  PRINT "WHAT DO YOU THINK THE"
305  PRINT "NEXT NUMBER SHOULD BE";
310  INPUT G
```

```
315  PRINT : PRINT : PRINT
320  IF G = A(T+1) THEN 360
330  PRINT "NO, THE NEXT NUMBER IS ";A(T+1);"."
335  PRINT : PRINT : PRINT
340  LET C = 0
350  GOTO 380
360  PRINT "THAT'S RIGHT!"
365  PRIT : PRINT : PRINT
370  LET C = C + 1
380  IF C = 3 THEN 420
390  LET T = T + 1
400  IF T < 10 THEN 240
410  GOTO 440
420  PRINT "YOU HAVE THE PATTERN.  GOOD WORK!"
430  PRINT : PRINT : PRINT
440  LET S = S + 1
450  IF S > N THEN 500
460  PRINT "LET'S TRY A NEW SET OF NUMBERS."
470  PRINT "TRY TO PREDICT THE NEXT NUMBER."
480  PRINT : PRINT : PRINT
490  GOTO 190
500  PRINT "THAT'S ALL FOR NOW.  THANKS FOR"
510  PRINT "INTERACTING WITH ME TODAY."
1000  DATA 2,4,6,8,10,12,14,16,18,20
1010  DATA 1,4,9,16,25,36,49,64,81,100
1020  DATA 1,2,4,7,11,16,22,29,37,46
9999  END

?RUN

I'M GOING TO SHOW YOU A SET OF NUMBERS.
I WANT YOU TO LOOK FOR A PATTERN AND
THEN TRY TO GUESS THE NEXT NUMBER
THAT FITS THE PATTERN.

THE FIRST 3 NUMBERS IN THE PATTERN ARE:

2,4,6

WHAT DO YOU THINK THE NEXT NUMBER SHOULD BE?8

THAT'S RIGHT!

THE FIRST 4 NUMBERS IN THE PATTERN ARE:

2,4,6,8

WHAT DO YOU THINK THE
NEXT NUMBER SHOULD BE?10

THAT'S RIGHT!

THE FIRST 5 NUMBERS IN THE PATTERN ARE:

2,4,6,8,10
```

```
WHAT DO YOU THINK THE
NEXT NUMBER SHOULD BE?12

THAT'S RIGHT!

YOU HAVE THE PATTERN.  GOOD WORK!

LET'S TRY A NEW SET OF NUMBERS.
TRY TO PREDICT THE NEXT NUMBER.
        .
        .
        .
        .

THAT'S ALL FOR NOW.  THANKS FOR
INTERACTING WITH ME TODAY.
```

Looking Back: Other number patterns are easily generated. Make up some of your own and try them with your students. Listed below are some of the examples of the wide range of possible sequences that can be used:

Multiples:	6,12,18,24,30,36,42,48,54,60
Arithmetic Sequences:	1,3,5,7,9,11,13,15,17,19
	8,5,2,-1,-4,-7,-10,-13,-16,-19
Geometric Sequences:	3,6,12,24,48,96,192,384,768,1536
	1,-2,4,-8,16,-32,64,-128,256,-512
Primes:	2,3,5,7,11,13,17,19,23,29
Composites:	4,6,8,9,10,12,14,15,16,18
Fibonacci Sequences:	1,1,2,3,5,8,13,21,34,55 4,2,6,8,14,22,36,58,94,152
Constant Sequences:	20,20,20,20,20,20,20,20,20,20
Triangular Numbers:	1,3,6,10,15,21,28,36,45,55

Topic: Math Logic

Problem: Recently a group of students had the pleasure of attending a performance by Math E. Magic, a mathematically inclined magician. He involved them in several interesting math magic tricks, the final one of which we want to share with you. He asked each member of the audience to choose a natural number and follow these instructions in order.

1. If your number is 1, stop.
2. If your number is even, divide it by 2 and then carry out instruction 1 with this new number.
3. If your number is odd, multiply it by 3, add 1 and then carry out instruction 1 with this new number.

When everyone was finished, he boasted that no matter what number is chosen his instructions will always produce a 1. No one in the audience could find a counterexample to the magician's claim. Can you?

Analysis: Remind students what a natural number is and then have them test the claim for a number of their choice. It will be helpful if they actually write the sequence of results. For example, if the number choice is 6, then the sequence of results is:

6, 3, 10, 5, 16, 8, 4, 2, 1 (stop).

Once students understand the instructions and their repetitive nature they can be guided to translate the sequence of instructions into a BASIC program.

Program: The BASIC program follows directly from the given

instructions. As such it can provide a nice introduction to the use of conditional branches (lines 70 and 80) and unconditional branches (lines 100 and 120) in programming. The program permits the user to easily test the claim for many cases, including large numbers.

```
10  REM  PROGRAM TO TEST
20  REM  MAGICIAN'S CLAIM
40  PRINT "ENTER A NATURAL NUMBER";
50  INPUT N
60  PRINT N;",";
70  IF N = 1 THEN 130
80  IF N / 2 = INT (N / 2) THEN 110
90  LET N = 3 * N + 1
100  GOTO 60
110  LET N = N / 2
120  GOTO 60
130  END

?RUN
ENTER A NATURAL NUMBER?10
10,5,16,8,4,2,1,

?RUN
ENTER A NATURAL NUMBER?13
13,40,20,10,5,16,8,4,2,1,

?RUN
ENTER A NATURAL NUMBER?128
128,64,32,16,8,4,2,1,
```

Looking Back: The results of our runs provide support for the magician's claim, but do not confirm (prove) it. In fact, the question of whether this algorithm always produced a "1" (i.e., terminates) remains an unsolved problem, despite the efforts of some of the best number theorists in the country. No one has provided a proof for the claim, nor has anyone found a counterexample. Perhaps one of your precocious students may find a solution and be recorded in history. Below are some related questions and problems all of your students can work on.

1 Explain how a computer could be used to prove the

magician's claim for all natural numbers less than or equal to 100. Modify the program to produce such a proof.

2 Let us define the **length** of a sequence as the number of terms in the sequence. The first program run produced a sequence of length 7. Can you find two consecutive natural numbers for which the magician's trick will produce the same length sequence? Can you find three consecutive natural numbers that will yield the same length sequences?

3 Is there a power of 2 which is generated first most frequently?

4 Explain how a computer might be used to **disprove** the magician's claim.

5 If the magician's claim were false we could run into an endless cycle in our program. To terminate execution we could use special terminal keys simultaneously - CTRL (Control) - C. If you are observing the output as it is being printed, what would be the signal the the program was not going to terminate? Modify the program so that it will automatically terminate if the magician's claim is false.

6 For variety, the magician is considering replacing instruction 3 with the following statement: "If your number is odd, multiply by 3, subtract 1 and then carry

out instruction 1 with this new number." Modify the program to test if the magician will be able to make the same claim if his trick is altered as described.

7 Investigate what would happen if instruction 3 were modified to "multiply by 5 and add 1" or "multiply by 5 and subtract 1."

Topic: Sieves

Problem: Mrs. Woods is on her way to the market to sell her eggs when a horseman knocks her down by accident, causing all of the eggs to be broken. He offers to pay for the damages and wants to know how many eggs she had. Mrs. Woods says that she doesn't remember the exact number, but when she took them two at a time, there was one left over. Also, there was one left over when they were taken 3, 4, 5, and 6 at a time. However, when they were taken 7 at a time, it came out evenly. What are the possible numbers representing the number of eggs that Mrs. Woods had?

Analysis: It is best to begin analysis of this problem by making a guess at the solution and checking to see if the guess fits the stated conditions. It becomes apparent immediately that a solution must be a multiple of seven and that one less than this multiple must be a common multiple of 2,3,4,5 and 6.

Program: Checking all multiples of seven less than 1000 (a restriction not stated in the problem) would be very time consuming. If the method of attack is clear to the students, a BASIC program should be written to carry out the needed computations. One program which will "sieve" out the multiples of seven which are not solutions to the problem is given below.

```
10  REM  PROGRAM TO SOLVE THE WOMAN
20  REM  AND THE EGGS PROBLEM USING
30  REM  A SIEVING TECHNIQUE.
60  PRINT  "THE POSSIBLE NUMBER OF EGGS:"
70  PRINT
80  FOR N = 7 TO 1000 STEP 7
100  REM  THE LOOP WHICH FOLLOWS SIEVES
110  REM  OUT NUMBERS WHICH ARE NOT
120  REM  COMMON MULTIPLES OF 2,3,4,5, & 6
140  FOR D = 2 TO 6
150  IF (N - 1) / D < > INT ((N - 1) / D) THEN 180
160  PRINT D
170  PRINT N,
180  NEXT N
190  END

?RUN
THE POSSIBLE NUMBER OF EGGS:

301          721
```

Looking Back: The printout from the sample program indicates two possible solutions less than 1000. Each of these should be verified by the students. Modifications of the program to make it more efficient are possible although discussions of such modifications should take place after the problem has been solved. (As an example, is it necessary to "sieve" out numbers by testing all of the divisors used here? If not, what is the minimal set of divisors that need be tested?)

This problem can be used with middle school or high school students when they are involved in using number theory. Assign it to middle school students while they are learning the divisibility rules. Use while they are listing factors in order to obtain the greatest common factor (GCF) for simplifying fractions or the least common multiple (LCM) for adding fractions. Assign to high school students who are using the GCF in factoring or in algebraic fractions. Some

additional problems which can be solved by using a sieving technique are given below.

1 Write a sieving program that will check whether a given positive integer is a prime number.

2 Find all of the prime numbers between 1 and 1000 using the Sieve of Eratosthenes.

3 Modify the program above to sieve out all the twin primes between 1 and 1000. Twin primes are prime numbers differing by two, such as 3 and 5.

Topic: Maximization

Problem: Mr. Lopez, a sheet metal worker, has been sub-contracted to manufacture rectangular storage bins for small automobile parts. He has been supplied with 30 x 16 cm. sheets of tin. The bins are to be made by cutting squares of the same size from each corner of a sheet and then bending up the "tabs" and spot-welding In order to obtain future manufacturing contracts he wants to make bins with the largest possible volume. What should the dimensions of the bins be?

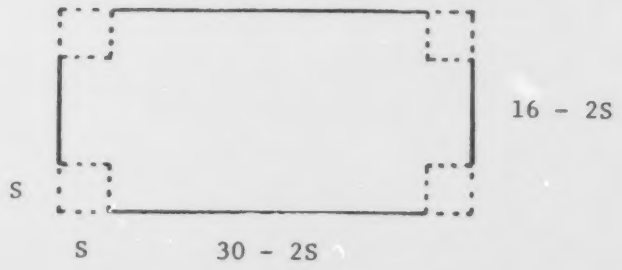
Analysis: To ensure that students understand the problem you may wish to take a piece of tagboard and illustrate the process by which the bins are to be made. Have them determine the volumes of the resulting boxes. In the process of this activity they need to make the following observations: If the square to be cut from each corner has length S, then

a. the height of the resulting bin is S

b. the largest possible value for S must be less than half the smallest dimension (i.e., $S < 8$)

c. the length will be $30 - 2S$

d. the width will be $16 - 2S$



Program: The above ideas can be easily translated into a BASIC program that initializes and then increments the length of the side of the square to be cut-out from each corner and, at each step computes and prints the resulting dimensions and volume of the storage bin. Note that line 180 checks the width each time, stopping the program when it becomes less than or equal to zero. Sample runs show results for squares with initial lengths and increments of 1.0 and 0.5 respectively. Analysis of the computed tabular results will suggest possible solutions to the problem.

```
10  REM  MAXIMIZING VOLUME
20  REM  OF AN OPEN BOX
40  PRINT "ENTER INITIAL LENGTH OF SQUARE CUT-OUTS"
50  INPUT S
60  PRINT
70  PRINT "HEIGHT", "WIDTH", "VOLUME"
80  FOR I = 1 TO 50
90  PRINT "-";
100  PRINT I
110  PRINT
120  REM  THIS LOOP ADJUSTS THE DIMENSIONS
130  REM  AND COMPUTES AND VOLUMES
150  FOR H = 2 TO 8 STEP S
160  LET L1 = 30 - 2 * H
170  LET W1 = 16 - 2 * H
180  LET W1 < = 0 THEN 220
190  LET V = L1 * W1 * H
200  PRINT H, L1,W1,V
210  NEXT H
220  END

?RUN
ENTER INITIAL LENGTH OF SQUARE CUT-OUTS
?1

HEIGHT          LENGTH          WIDTH           VOLUME
-------------------------------------------------------
1               28              14              392
2               26              12              624
3               24              10              720
4               22              8               704
5               20              6               600
6               18              4               432
7               16              2               224
```

```
?RUN
ENTER INITIAL LENGTH OF SQUARE CUT-OUTS
?.5
```

HEIGHT	LENGTH	WIDTH	VOLUME
.5	29	15	217.5
1	28	14	392
1.5	27	13	526.5
2	26	12	624
2.5	25	11	687.5
3	24	10	720
3.5	23	9	724.5
4	22	8	704
4.5	21	7	661.5
5	20	6	600
5.5	19	5	522.5
6	18	4	432
6.5	17	3	331.5
7	16	2	224
7.5	15	1	112.5

Looking Back: This problem can be used with students from upper elementary to high school level whenever they are working with areas and volumes. It is important to run the program for two or more height increments (side lengths of square cut-outs) to obtain better approximations. 720 cm^3 is the maximum volume when the height is incremented by 1 cm whereas 724.5 cm^3 is the maximum volume when the height is incremented by 0.5 cm. Better approximations can be found by incrementing heights by 0.2 or 0.1 cm. By using different constants in lines 150, 160, and 170 pupils can maximize volumes of other open-top rectangular boxes to be constructed from sheets of material in the same way.

Depending on the level of your students you may wish to have them pursue one or more of these related activities.

1. Suppose Mr. Lopez was supplied with sheets having dimensions 20 x 24 cm. Note that the area of these sheets is the same as that of the sheets in the problem. Modify the program to use this data and compare the tabular results.
2. Modify the program so that the user can input the dimensions of the initial sheets of tin.
3. Modify the given program or the one written for exercise 2 so that it also computes and prints the surface area of the box.
4. Write a program to minimize surface area, given a specific volume of 726 cm^3.
5. A food packing company, making its own cans, wishes to pack 512 cm^3 of liquid into a can. What is the minimum amount of metal that can possibly be used?
6. Modify problem 5 by using 354 cm^2 of metal (surface area) and then find the maximum volume.

Topic: Probability

Problem: Twenty-three people are sitting in a large waiting room. Having nothing more productive to do, they decide to compare birthdays (month and day of month). Determine the probability that two or more of the twenty-three people will have the same birthday.

Analysis: To be certain everyone understands the problem, it is a good idea of have your students compare birthdays before attempting to solve the problem.

The birthday problem is difficult to solve directly. It is much easier to find the probability of the complementary event: all 23 people have different birthdays. The desired probability can then be found by subtracting this result from 1.

For purposes of analysis, let each day of the year be represented by a box labeled with one of the numbers 1 through 365 (ignore leap years). Each of the 23 people is to place a slip of paper into the box corresponding to his/her birthday. We want to find the probability that none of the boxes will have more than one slip inside.

The first person can place a slip into any one of the 365 boxes and not cause a box to contain more than one slip. Therefore the probability that the first person will place a slip in an unused box is 365/365 or 1.

Once the slip is placed, the second person can place a slip into any one of the 364 remaining boxes and not cause duplication. This probability that the second person will select one of these boxes is 364/365.

Assuming the first two have used different boxes, the third person can place a slip into any one of the 363 boxes and not cause duplication. The probability of this occurring is 363/365.

Continuing this reasoning, and assuming the twenty-three birthdays are independent of each other, the probability that all 23 people will place slips into different boxes (i.e., have different birthdays) is given by:

365/365 x 364/365 x 363/365 x . . . x 343/365.

It follows that the probability of the complementary event, at least two people having the same birthday, is given by:

1 - 365/365 x 364/365 x 363/365 x . . . x 343/365.

Program: The above computations would be very tedious, even with a hand calculator available. A BASIC program can easily be written to perform the needed computations. A sample listing and RUN are given below.

```
10  REM  PROGRAM TO COMPUTE PROBABILITY
20  REM       FOR BIRTHDAY PROBLEM
40  LET P = 1
50  FOR I = 1 TO 23
60  LET P = P * (365 - I + 1) / 365
70  LET I
80  LET Q = 1 - P
90  PRINT
100  PRINT "FOR 23 PERSONS, THE PROBABILITY"
110  PRINT "THAT AT LEAST TWO HAVE THE SAME"
120  PRINT "BIRTHDAY IS ";Q;"."
130  END
```

```
?RUN

FOR 23 PERSONS, THE PROBABILITY
THAT AT LEAST TWO HAVE THE SAME
BIRTHDAY IS .507297234.
```

Looking Back: The program results indicate that for a group of 23 people, the probability that two or more will have the same birthday is greater than 1/2.

Once the BASIC program has been written and the original problem solved, several modifications of the problem could be investigated by making minor changes in the program.

1. What is the probability that at least two members of your mathematics class have the same birthday?
2. Modify the program so that it will compute the probability for a room of n people where the user inputs the value of n.
3. Modify the program to print the probabilities for 5, 10, 15, . . ., and 60 people. Graph the probabilities on a coordinate system and study the results.
   a. How large a group of people is necessary for the probability to be greater than .00?
   b. How large a group is required for the probability to be 1.00?
4. In our analysis of the birthday problem we used a 365-day year, ignoring February 29. Write a BASIC program that will provide a solution to the birthday problem taking into account the existence of leap years.

Problem Solving Using Simulation Method

Topic: Resource Management

Problem: In 1830 there were 40 million buffalo in the western United States. By 1887 only 200 buffalo remained because of a lack of policy for resource management in this country. Today there are approximately 26,000 buffalo distributed as follows:

adult males - 10,400	male calves - 3,380
adult females - 9,100	female calves - 3,120

The Department of the Interior is considering the implementation of a controlled harvesting program on these animals. They first need to formulate a policy on how many buffalo can be taken each year. It has been decided that no calves or adult females may be harvested. The department is considering allowing the harvesting of 1,000 adult males annually. What will be the effect of this harvesting policy on herd size over the next 10 years?

Analysis: In order to solve this problem we must first develop a model of population growth for the buffalo herd. Recent studies provide the following facts:

1. Calves are considered adult animals at age 2.
2. For every 100 adult females at the beginning of a year, 90 calves are born during the year of which 48 are male and 42 are female.
3. Only 50 percent of the calves reach one year of age and of these 60 percent reach maturity. The survival rate for adults is 90 percent.

Elementary algebra can now be used to construct a mathematical model. Let us use M and F to denote the current numbers of adult males and adult females respectively. Initially M = 10,400 and F = 9,100.

Without additional information about the age distribution of the calves we will assume two-thirds of the males and two-thirds of the females are newborns and denote these numbers by M1 and F1 respectively. Thus M1 = 2/3 x 3,380 and F1 = 2/3 x 3,120. We will assume the remaining calves are one year of age and denote the numbers of these males and females by M2 and F2 respectively. At the beginning of the period one year from now the buffalo population can be described as follows:

a. new male calves (M1): .48F

b. new female calves (F1): .42F

c. one year old male calves (M2): .5M1

d. one year old female calves (F2): .5F1

e. adult males (M): .90M + .6M2 - 1000

f. adult females (F): .90F + .6F2

yielding a herd size of M + F + M1 + F1 + M2 + F2.

Program: Now that we have a mathematical model for describing the buffalo population we can write a BASIC program that uses the model to simulate changes in the herd size over time - in our case over the next 10 years. Initially one might think that the replacements for the variables suggested above could be directly translated into BASIC statements. However, the computer executes assignments sequentially, one by one, rather than simultaneously. As soon as the replacement suggested in (a) is executed, the old value of M1 is lost. But it is needed for the replacement suggested in (c). Note that replacements

(c) and (d) cannot be executed first since then the old values of M2 and F2 would not be available for use in (e) and (f). To eliminate this problem we make copies of the values of M,F,M1,F1, M2, and F2 using the variables T1, T2, T3, T4, T5, T6 respectively.

The following program prints a table of the buffalo population distribution for the next 10 years, beginning with the present population.

```
10  REM  SIMULATION PROGRAM FOR
20  REM   BUFFALO POPULATION
40  PRINT  "ENTER NUMBER OF ADULT MALES";
50  INPUT M
60  PRINT
70  PRINT "ENTER NUMBER OF ADULT FEMALES";
80  INPUT F
90  PRINT
100  PRINT "ENTER THE NUMBER OF MALE CALVES";
110  INPUT C1
120  PRINT
130  PRINT "ENTER THE NUMBER OF FEMALE CALVES";
140  INPUT C2
150  REM  MODEL ASSUMES TWO-THIRD OF THE
160  REM  CALVES ARE NEWBORN AND THE OTHERS
170  REM  ARE 1 YEAR OLD
180  LET M1 = (2 / 3) * C1
190  LET M2 = C1 - M1
200  LET F1 = (2 / 3) * C2
210  LET F2 = C2 - F1
220  PRINT
230  PRINT "BUFFALO POPULATION DISTRIBUTION:
240  PRINT "HERD", "ADULT", "ADULT", "MALE", "FEMALE"
250  PRINT "SIZE", "MALES", "FEMALES", "CALVES", "CALVES"
260  FOR I = 1 TO 72
270  PRINT "-";
280  NEXT I
290  PRINT
300  FOR Y = 1 TO 11
310  LET S = M + F + M1 + F1 + M2 + F2
320  PRINT  INT (S + .5), INT (M + .5), INT (F + .5)
     INT (M1 + M2 + .5), INT (F1 + F2 + .5)
330  LET T1 = M
340  LET T2 = F
350  LET T3 = M1
360  LET T4 = F1
370  LET T5 = M2
380  LET T6 = F2
390  LET M = .9 * T1 + .6 * T5 - 1000
400  LET F = .9 * T2 + .6 * T6
```

```
410  LET M1 = .48 * T2
420  LET F1 = .42 * T2
430  LET M2 = .5 * T3
440  LET F2 = .5 * T4
450  NEXT Y
460  END

?RUN
ENTER NUMBER OF ADULT MALES?10400

ENTER NUMBER OF ADULT FEMALES?9100

ENTER THE NUMBER OF MALE CALVES?3380

ENTER THE NUMBER OF FEMALE CALVES?3120
```

BUFFALO POPULATION DISTRIBUTION

HERD SIZE	ADULT MALES	ADULT FEMALES	MALE CALVES	FEMALE CALVES
26000	10400	9100	3380	3120
28207	9036	8814	5495	4862
28393	7808	8557	6415	5613
27853	7338	8848	6223	5445
27760	6873	9073	6300	5513
27810	6418	9244	6479	5669
27888	6050	9435	6615	5788
28037	5752	9634	6747	5904
28260	5508	9836	6889	6028
28544	5316	10041	7033	6154
28885	5171	10251	7180	6283

Looking Back: The results of our simulation indicate the effects of the proposed harvesting policy over the next 10 years. The obvious advantage of simulations is that we can examine the effects of harvesting policies without actually endangering the buffalo population. The results of the simulations may be used to establish logical resource management decisions.

The mathematical model could be discussed in general terms with middle school pupils who could then use this program to examine the effects of different harvesting policies. With high school students you would work through a derivation of the model with them and assign

the development of the simulation program. Related activities and problems students might investigate include:

1. Could one avoid the use of the temporary variables T1 - T6 by sequencing the replacements as follows: e,f,c,d,a,b? Explain?
2. Modify the program so the user can input the size of the adult male harvest and then investigate the effects of differing harvest sizes on the herd size over a 10 year period.
3. Modify the program in exercise 2 so that it also permits the harvesting of adult females and examine the effects of this new policy on herd size over a period of years.
4. Try to find a constant harvesting policy for adult males and females that will maintain the herd at its present level over the next 5 years.
5. a) Write to the Michigan Department of Natural Resources to obtain data on the population distribution, birth-rate, survival rates, and harvesting policy for deer in Michigan. Construct a mathematical model for the deer population. (Assume that fawns reach maturity at one year of age). Develop a simulation program to investigate the effects of alternate harvesting policies on this important Michigan resource.

   b) Read about "Life or Death for the Harp Seal" in the January, 1976 issue of National Geographic. Will this valuable marine resource survive present levels of harvesting or will it become extinct? Write to the Canadian Minister of Fisheries for appropriate

data from which you can develop a mathematical model and simulation program for investigating policies that will save the seals.

Topic: Number Theory

Problem: A small school has a hall which contains 100 lockers. The lockers are labeled with the numbers 1 through 100 and are all closed. One hundred students are outside the school and plan to enter the school one at a time. The first student will run down the hall and open every locker. The second student will then run down the hall and close every lock with an even number. The third student will run down the hall and change the state of every locker labeled with a multiple of 3 (opening those that are closed and closing those that are open). The fourth student will change the state of all lockers that are labeled with a multiple of 4. This process will continue until all 100 students have entered the school and changed the state of the lockers according to the plan. Which lockers will be left open at the end of this process?

Analysis: To be certain everyone understands the problem, it is best to work through the process with a smaller number of lockers and students, say six of each. The table below summarizes the movement of six students in a hall containing six lockers. A "0" is used to indicate a closed locker and a "1" is used to indicate an open locker.

	Locker Number					
	1	2	3	4	5	6
Original State	0	0	0	0	0	0
After 1st Student	1	1	1	1	1	1
After 2nd Student	1	0	1	0	1	0
After 3rd Student	1	0	0	0	1	1
After 4th Student	1	0	0	1	1	1
After 5th Student	1	0	0	1	0	1
After 6th Student	1	0	0	1	0	0

For six lockers and six students the process leaves lockers numbered

"1" and "4" open and all others closed.

Program: Carrying out the process for all 100 lockers would be a long and boring task. The computer program below simulates the process described in the problem and prints and numbers of the lockers left open at the end of the process. In an upper elementary or junior high class, the program could be provided by the teacher and discussed before execution. More advanced students could be asked to write the program to solve this problem.

```
10  REM  PROGRAM TO SIMULATE THE
20  REM    100 LOCKER PROBLEM
40  DIM L(100)
50  REM  INITIALIZE EACH LOCKER TO STATE 0
60  REM  TO INDICATE THAT IT IS CLOSED
90  FOR I = 1 TO 100
95  LET L(I) = 0
100  NEXT I
105  REM  100 STUDENTS RUN DOWN THE ROW
110  REM  OF LOCKERS CHANGING THE STATE
115  REM  OF EACH AS DESCRIBED IN THE
120  REM  PROBLEM.  EACH STUDENT S STARTS
125  REM  AT LOCKER S.
130  REM     L(I)=0 -->LOCKER I IS CLOSED
135  REM     L(I)=1 -->LOCKER I IS OPEN.
140  FOR S = 1 TO 100
145  FOR I = S TO 100 STEP S
150  IF L(I) = 0 THEN 180
160  LET L(I) = 0
170  GOTO 190
180  LET L(I) = 1
190  NEXT I
200  NEXT S
210  PRINT
230  PRINT "THE FOLLOWING LOCKERS ARE OPEN:"
240  PRINT
250  FOR I = 1 TO 100
260  IF L(I) = 0 THEN 280
270  PRINT I,
280  NEXT I
290  END

?RUN

THE FOLLOWING LOCKERS ARE OPEN:

1        4        9        16       25

36       49       64       81       100
```

Looking Back: Follow-up questions that might be considered are given below.

1 If we had started with 1000 lockers and 1000 students, what lockers would be open at the end of the process? Predict the results and then modify the program to verify your prediction.

2 The list of lockers remaining open after the process was completed contained all of the perfect square numbers less than or equal to 100. Explain why only these particular lockers remained open. (Hint: Consider the factors of each locker number.)

3 If we had started with 10,000 lockers and 10,000 students, would the last locker be open or closed? Would locker 4,964 be open or closed? (Do not use the computer program to answer these.)

Topic: Probability

Problem: What is the probability that on the starting lineup of any given basketball team, at least 2 players were born in the same month?

Analysis: Students who have studied some probability may recognize this problem as similar to the birthday problem solved in the first section of this chapter and thus may be solved by a similar analytic technique. However, an estimate of the probability can be obtained even by students with little formal study of the probability by using Monte Carlo methods.

As a model for this problem situation we can represent the months January through December by the numbers 1 through 12 respectively. For our purposes we can simulate the birthmonths of 5 starting players on a team by rolling an icosahedral die 5 times or by selecting 5 numbers between 1 and 12 from a table of random numbers. Such a trial will be called successful if a number is obtained more than once. By repeating this simulation for 50 or 100 trials we can obtain an estimate of the solution for our problem by computing the ratio:

$$\frac{\text{number of successful trials}}{\text{total number of trials}}$$

As the number of trials is increased, this ratio approaches the exact probability.

Program: The following program is designed to carry out the simulation

model as described with the user entering the desired number of trials. For each trial the program generates 5 random numbers between 1 and 12 and then checks if there is a duplication. If so, the trial is counted as a success. After the specified number of trials have been conducted, the probability ratio is computed and printed as an estimate of the solution for the problem. (Note that simulation programs such as this one and those which follow that involve the generation of random numbers will take a few minutes to execute on microcomputers.)

```
10  REM  PROGRAM TO SIMULATE THE
20  REM   BIRTHMONTH PROBLEM
40  DIM (F12)
50  PRINT "ENTER THE NUMBER OF TRIALS DESIRED";
60  INPUT N
70  LET S = 0
80  FOR K = 1 TO N
90  REM  SET BIRTHMONTH FREQUENCIES TO ZERO
110  FOR J = 1 TO 12
120  LET F(J) = 0
130  NEXT J
140  REM  GENERATE 5 RANDOM BIRTHDAYS
160  FOR I = 1 TO 5
170  LET B = INT (12 * RND (1)) + 1
180  LET F(B) = F(B) + 1
190  NEXT
200  REM  CHECK FOR DUPLICATION OF BIRTHMONTHS
220  FOR H = 1 TO 12
230  IF F(H) > 1 THEN 260
240  NEXT H
250  GOTO 270
260  S = S + 1
270  NEXT K
280  PRINT
290  REM  COMPUTE AND PRINT PROBABILITY
310  PRINT "FOR 5 PEOPLE, THE PROBABILITY THAT"
320  PRINT "AT LEAST 2 PEOPLE HAVE THE SAME BIRTHMONTH IS"; S / N
330  END

?RUN
ENTER THE NUMBER OF TRIALS DESIRED?50

FOR 5 PEOPLE, THE PROBABILITY THAT
AT LEAST 2 PEOPLE HAVE THE SAME BIRTHMONTH IS .6

?RUN
ENTER THE NUMBER OF TRIALS DESIRED?100
```

```
FOR 5 PEOPLE, THE PROBABILITY THAT
AT LEAST 2 PEOPLE HAVE THE SAME BIRTHMONTH IS .72

?RUN
ENTER THE NUMBER OF TRIALS DESIRED?100

FOR 5 PEOPLE, THE PROBABILITY THAT
AT LEAST 2 PEOPLE HAVE THE SAME BIRTHMONTH IS .54
```

Looking Back: Observing the program output for several runs indicates differing probabilities. Discuss reasons for this with pupils and perhaps suggest averaging the results. Depending on the background of your students you might use analytic methods to show the probability that at least 2 persons in a group of 5 picked at random have the same birthmonth is

$$1 - \left(\frac{12}{12}\right)\left(\frac{11}{12}\right)\left(\frac{10}{12}\right)\left(\frac{9}{12}\right)\left(\frac{8}{12}\right) = 1 - \frac{95040}{248832} \doteq .62$$

Students in elementary school through college find enjoyment and success in solving probability problems by Monte Carlo methods. You might consider additional tasks and problems similar to those below.

1 Is it possible to obtain the exact solution for the birth-month problem using Monte Carlo methods?

2 Modify the program presented so that it will compute the probability for a group of n people, where the user inputs the value of n.

3 Modify the program so that it will compute an estimate of the probability for the birthday problem on page 82. Compare your estimate with the theoretical value.

4 The program presented here could be made more efficient by testing for a match as the random numbers are generated. Modify the program so that the FOR-NEXT loop with index H can be omitted.

5 Modify the program so that it will compute the probability that at least 3 of a group of n people chosen at random have the same birthmonth (birthday).

6 Prior consumer experience indicates that 5 percent of the light bulbs produced in a certain factory are defective. If you purchase 6 of these bulbs from a store, what is the probability of getting 2 or more bad bulbs? Use Monte Carlo methods to obtain a solution.

Topic: Expected Value

Problem: A cereal company plans to introduce a new sugar-coated cereal to the buying public. To encourage shoppers to purchase their new product they are enclosing a free coloring pen in each box of cereal. The pens are available in six different colors (red, blue, yellow, green, brown, and orange). They decide to randomly distribute equal numbers of each of the six pens in their first production of cereal.

If you (as a typical customer) would like to collect all six pens, how many boxes of cereal would you expect to buy?

Analysis: One way to answer the question is to go to the grocery store and buy boxes of cereal until a pen of each color has been collected. A better estimate of the expected (average) number of boxes needed could be found by having several people buy cereal until each collected the pens and then find the average number of boxes purchased. Since buying cereal in such quantities is impractical, we need a way to simulate the purchasing of the cereal.

One of the easiest ways to simulate this situation is to represent the different pen colors with six numbers (1 = red, 2 = blue, 3 = yellow, 4 = green, 5 = brown, 6 = orange) and then randomly generate a sequence of these numbers until we get at least one of each number. The length of this sequence represents the number of boxes of cereal that had to be purchased (in that trial) before the entire pen set was collected. A sequence of random numbers ranging from 1 through 6 is easily generated by tossing a single die several times and recording

the outcomes. Teachers should have their students simulate this situation several times using a standard die. A record of the outcomes for each trial could be kept in a table similar to the one below.

Pen Color

Trial	Red 1	Blue 2	Yellow 3	Green 4	Brown 5	Orange 6	Boxes Purchased
1							
2							
3							

The average of a few trials normally gives a good approximation to the expected value, especially when student trials are pooled across the class.

Program: The chances of obtaining a more accurate estimate to the solution are increased by averaging the results from an even larger number of trials. The computer can easily be programmed to simulate many trials.

The BASIC program below simulates the cereal problem by randomly generating integer values from 1 to 6 until at least one of each number has been generated. The user of the program can input the number of trials he/she wishes to simulate. The output from the program gives the average number of boxes purchased over all trials. This average is an estimate of the true expected value for this problem.

```
10  REM  PROGRAM SIMULATES PURCHASING BOXES
20  REM  OF CEREAL.  EACH BOX CONTAINS ONE
30  REM  OF SIX DIFFERENT COLORED PENS.  THE
```

```
40  REM  AVERAGE NUMBER OF BOXES REQUIRED
50  REM  TO COLLECT ALL SIX PENS IS COMPUTED
60  REM  AND PRINTED.
90  REM  VARIABLES USED IN PROGRAM:
100  REM  N-NUMBER OF TRIALS (INPUT)
110  REM  T-TOTAL BOXES PURCHASED DURING A TRIAL
120  REM  C-CUMULATIVE TOTAL FOR T OVER ALL TRIALS
130  REM  P-PEN COLOR RECEIVED IN BOX (1<=P<=6)
140  REM  F(P)-FREQUENCY OF PEN COLOR DURING TRIAL
200  PRINT
210  PRINT "ENTER THE NUMBER OF TRIALS DESIRED"
220  INPUT N
225  LET C = 0
230  FOR I = 1 TO N
240  FOR J = 1 TO 6
250  LET F(J) = 0
260  NEXT J
270  LET T = 0
280  LET P = INT (6 * RND (1)) + 1
290  LET F(P) = F(P) + 1
300  LET T = T + 1
310  FOR J = 1 TO 6
320  IF F(J) = 0 THEN 280
330  NEXT J
340  LET C = C + T
350  NEXT I
360  PRINT : PRINT : PRINT
370  PRINT "THE AVERAGE NUMBER OF BOXES PURCHASED"
380  PRINT "TO COLLECT ALL SIX PENS WAS"; C / N
400  END

?RUN

ENTER THE NUMBER OF TRIALS DESIRED
?100

THE AVERAGE NUMBER OF BOXES PURCHASED
TO COLLECT ALL SIX PENS WAS 14.29

?RUN

ENTER THE NUMBER OF TRIALS DESIRED
?100

THE AVERAGE NUMBER OF BOXES PURCHASED
TO COLLECT ALL SIX PENS WAS 16.08
```

Looking Back: The cereal problem can easily be understood by upper elementary, junior high, and senior high students. You may wish to indicate that the theoretical expected value for the six pens cereal problem is 14.7. The computer simulation was executed twice with 100

trials on each execution. One of the estimates was below the theoretical value and one was above. These results indicate the approximate nature of solutions generated by Monte Carlo techniques. Variations of the original problem and an additional problem can be used as follow-up:

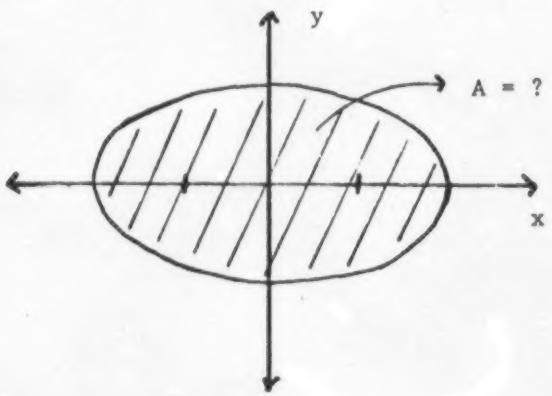
1 What is the expected value if eight differently colored pens instead of six are used? Ten pens? Twelve pens? (These possibilities can be simulated physically by using a polyhedral die.)

2 Modify the computer program to simulate the cases mentioned in problem 1.

3 A recreational diversion consists of two towers built of 3 and 7 blocks respectively. A tower is chosen at random and then a block is removed from this tower and placed on the top of the other tower. This procedure is repeated until all of the blocks have been moved from one tower to the other. On the average, about how many moves are required to move the blocks from one tower to the other? (To choose a tower at random you can flip a coin or simulate flipping the coin with the BASIC statement LET C = INT(2 * RND(1)) + 1).

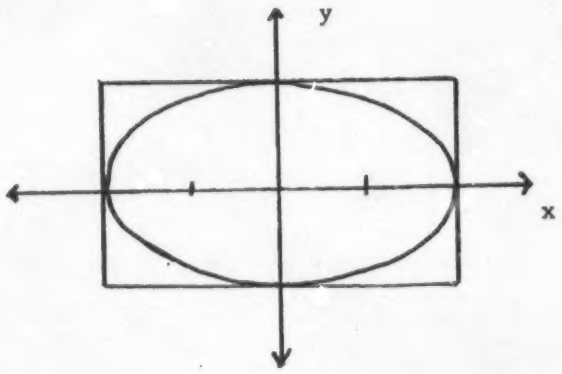
Topic: Approximating Areas

Problem: We all know formulas for finding areas of special regions such as those bounded by circles, triangles, or parallelograms, but how can we find the area of regions for which we do not know a formula? For example, how could we find the area of the region bounded by the ellipse $\frac{x^2}{4} + y^2 = 1$.



Analysis: One method of finding the area would be to subdivide, as closely as possible, the region into several smaller regions for which area formulas are known. The sum of the smaller areas would serve as an approximation to the area of the ellipse. This approach is intuitively sensible and is familiar to anyone who has been exposed to a semester of integral calculus.

Another method that is easier to understand and more fun to carry out is that of dart throwing! First enclose the ellipse in a rectangle as shown below. The area of the rectangle is 8 square units. The area of the ellipse is therefore less than 8 square units.



If you were to use the rectangle as a dart board and randomly toss several darts into the board, some of the darts would land inside (or on) the ellipse and some would not. If your dart throwing were truly random, you would expect the following proportion to hold:

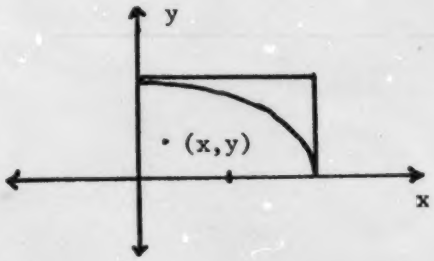
$$\frac{\text{Area of Ellipse}}{\text{Area of Rectangle}} = \frac{\text{Number of Darts Hitting Ellipse}}{\text{Number of Darts Thrown}}$$

Since the only unknown quantity in this proportion is the area of the ellipse, you could solve the proportion and get an approximate measure of that area. Rather than wearing your arm out throwing darts, why not write a computer program to simulate the dart throwing activity?

To simplify the programming we can use the symmetry of the situation and use only the first quadrant to compute the ratio of darts hitting the ellipse to total darts thrown. Any dart thrown must land at some ordered pair (x,y) within the rectangle enclosing the ellipse.

Thus $0 \leq x \leq 2$ and $0 \leq y \leq 1$.

We can simulate a thrown dart by randomly generating values of x and y within this range. Once a dart is thrown, it must be determined if it has landed within the ellipse or not. If $x^2/4 + y^2 \leq 1$ then the dart (the ordered pair (x,y)) has landed within (or on) the ellipse. If a total of N darts are thrown and H of the darts hit within the ellipse, the approximate area (A) of the ellipse can be found by solving the proportion A/8 = H/N. Therefore, A = 8 x H/N.



Program: The following program simulates the throwing of random darts and uses the algorithm described above to approximate the area of the region bounded by the ellipse $x^2/4 + y^2 = 1$.

```
10  REM  PROGRAM TO APPROXIMATE THE AREA
20  REM  OF THE ELLIPSE X^2/4+Y^2=1
30  REM  BY TOSSING RANDOM DARTS.
40  PRINT "INPUT THE NUMBER OF DARTS YOU WISH TO THROW"
50  INPUT N
60  LET H = 0
70  FOR T = 1 TO N
80  LET X = 2 * RND (1)
90  LET Y + RND (1)
100  LET D = X ^ 2 / 4 + Y ^ 2
110  IF D > 1 THEN 130
120  LET H = H + 1
130  NEXT T
140  LET A = 8 * H / N
150  PRINT : PRINT
170  PRINT H:" OF THE ";N;" DARTS"
180  PRINT "LANDED IN THE ELLIPSE."
190  PRINT
200  PRINT  "THE APPROXIMATE AREA OF"
210  PRINT  "THE ELLIPSE IS ";A;" SQUARE UNITS.
220  END

?RUN
```

```
INPUT THE NUMBER OF DARTS YOU WISH TO THROW
?1000

782 OF THE 1000 DARTS
LANDED IN THE ELLIPSE.

THE APPROXIMATE AREA OF THE
ELLIPSE IS 6.256 SQUARE UNITS.

?RUN
INPUT THE NUMBER OF DARTS YOU WISH TO THROW
?1000

785 OF THE 1000 DARTS
LANDED IN THE ELLIPSE.

THE APPROXIMATE AREA OF THE
ELLIPSE IS 6.28 SQUARE UNITS.
```

Looking Back: The program was executed twice with 1000 darts thrown on each simulation. Note the approximation nature of the solutions since slightly different results were obtained on each execution. (The actual area of the ellipse is 2π square units.)

The program presented here simulated all of the dart throwing activity and performed all of the necessary computations. Junior high math students would not be familiar with the equation of an ellipse. These students could be provided with a drawing of the ellipse and led to a discussion of the dart throwing activity. The ellipse could then be graphed and the enclosing rectangle constructed. A computer program could be written to generate ordered pairs in the needed range. Students could plot these ordered pairs on their graphs, count the number of hits, and solve the necessary proportion. Those who are skeptical of the accuracy of this method should use the simulation to approximate the area of a region for which the exact area is known. For example, the area of a unit circle could be approximated. Since this area is equal to the irrational number π, such a simulation

would serve as a method for approximating the decimal value of π.

Once convinced of the usefulness of this method, students could use it to approximate areas under curves defined by functions such as:

a) $f(x) = \{(x,y) | y = 3x + 2, 0 \leq x \leq 5\}$ and bounded by the line $x = 5$ and the positive x-axis. (Compare your answer with the actual area.)

b) $g(x) = \{(x,y) | y = x2, 0 \leq x \leq 10\}$ and bounded by $x = 10$ and the positive x-axis.

c) $h(x) = \{(x,y) | y = \sin(x), 0 \leq x \leq \pi\}$ and bounded by the positive x-axis and the line $x = \pi$.

Pupils in advanced mathematics courses might also be encouraged to use a similar method to approximate the volume of the solid bounded by the surface defined by $x^2 + y^2 + z^2 + 8x - 8y - 2z = 16$. (Compare your answer with the actual volume.)

References

Comprehensive School Mathematics Program. Chapter 16, Introduction to Computer Programming. Elements of Mathematics, Book 0: Intuitive Background. St. Ann, MO: CEMREL Inc., 1971.

Nierengelt, J., Farrar, J., & Reingold, E. Computer Approaches to Mathematical Problems. Englewood Cliffs, N.J.: Prentice-Hall, 1974.

Travers, K. & Gray, K. The Monte Carlo Method: A Fresh Approach to Teaching Probabilistic Concepts. The Mathematics Teacher. 74 (May, 1981), pp. 327-334.

CHAPTER SIX

DRILL, PRACTICE AND TUTORIAL USES OF COMPUTERS

Art Davidge
Pioneer High School
Ann Arbor, Michigan

Drill and practice (DP), that is using the computer to generate exercises for student solution, is one of the oldest and most widely used examples of Computer Assisted Instruction. The advantages of DP include: individualized, random or predefined exercise sets can be given to each student; immediate feedback can be given; and automatic record keeping facilities can be included. There are many forms of DP, ranging from those that entail a complete curriculum with built-in management functions to simple programs that generate the same type of exercise over and over again.

When DP includes diagnostic sections, initial instruction sequences, instruction review sequences based on student responses, and prescriptive messages to the student or teacher, the term TUTORIAL program is used. A good tutorial program must anticipate a multitude of student responses, must include appropriate branching routines which depend on responses, and certainly must reflect effective instruction.

Often, special authoring languages like COURSEWRITER, TUTOR, and PILOT are used to create tutorial programs. Tutorial applications require large numbers of computers or terminals, sophisticated software, and relatively little initial teacher training for implementation. See Kleiman and Humphrey (1982) for an overview of using authoring languages to create tutorial programs.

This chapter will give an overview of an extensive DP program and focus on techniques teachers can use for designing short, customized DP programs for particular content and students.

An example of a student interaction with an extensive DP package program is shown in the following sequence. Note that in actual practice, each student has an ID number and is automatically given the next lesson in sequence. In the following example, the student's responses are underlined.

START

What is your ID number and first name?1027,JOE

Is your last name COOL?YES

Course name?$M2

21 March 1981 13:54 Port 11

HELLO, JOE. WE HOPE YOU ENJOY TODAY'S PROBLEMS.

M 3153

********** HERE WE GO ! ! ! ! ! **********

+ 3 = 1

-2

(-3) X = (-36)

12

(-12) X = (-48)

2

WRONG, TRY AGAIN

(-12) X = (-48)

4

LESSON OVER. YOU ANSWERED 10 OUT OR 12 QUESTIONS CORRECTLY.

HERE COMES YOUR NEXT LESSON, JOE.

M 3153

(-9) + (-5) =

TIME IS UP TRY AGAIN

(-9) + (-5) =

-14

(-8) + 2 =

-6

5 X (-9) =

-45

(-9) X 3 =

//STOP

GOODBYE JOE.

PROGRAMS PROPERTY OF HEWLETT-PACKARD CO. ALL RIGHTS RESERVED.

An entire individualized arithmetic curriculum can be augmented by a package of this type. The student is moved through the various

strands or blocks depending on the percentage of correct answers and his/her current placement.

Periodic reviews are automatically scheduled. The teacher has the opportunity to vary almost all parameters involved in the lessons. These include: response time, length of each session, and actual placement of the student in the curriculum. Also available are many different reports that the teacher can obtain at any time. These include: total usage time for individual students or classes, highlights of low or high performance by students on particular objectives, notification when students will meet new content, and general class and student record keeping reports.

A comprehensive DP package generally requires vast storage capacities and thus are most commonly found on large time-sharing systems. If a school system has available the resources of an intermediate school district (ISD) time-share system, programs such as the one from which the example above was taken are readily available. Terminals and phone charges are required to use these systems. A few multi-disk packages have been produced for microcomputers, but their storage capacities are generally too limited for comprehensive packages. Microcomputers can, however, act as terminals for time-share programs. A great advantage of using a DP package is that no programming knowledge is required of the teacher or student. Teachers can use the package with little formal training or computer experience.

There are often times when a teacher needs a specific type of

exercise, lasting for a relatively short time to give students practice or reinforcement in specific content. The remainder of this chapter will illustrate programming methods useful in generating practice exercises, keeping track of responses, and reporting results. It should be noted that students in a computer programming course may be asked to write these programs. This has several advantages, including: the student gets practice in BASIC programming; the student must understand the algorithms involved, including special cases; and the teacher can acquire a wide array of programs with minimal investment of time. Many high school teachers cooperate with elementary or junior high teachers in providing quality student written programs that younger students can use. The key to an arrangement of this type is the careful specification of program requirements by the requesting teacher.

Familiarity with the BASIC statements: LET, PRINT, IF/THEN, READ and DATA, and FOR/NEXT is assumed in the following examples. See Chapter Three for a review of these BASIC statements.

At the heart of any DP program is a built-in function known as the random number generator. Most versions of BASIC employ a random number generator which generates a random number between .000000 and 1.000000. In most cases this interval is closed below and open above, (i.e., 0 is included as a possibility but 1 is not). The number of decimal places included is dependent on the particular machine used. Usually a number such as 1 or 0 must be used as a "seed", hence:

```
10  LET A=RND(1)
```

generates a random number in the range indicated above and stores the

random number in the variable A. When used with the greatest integer function (INT), any range of random digits can be generated:

```
10  LET R=INT(RND(1)*(1+1))          between 0 and A
10  LET R=INT(RND(1)*(A+1))+1        between 1 and A
10  LET R=INT(RND(1)*(B-A+1))+A      between A and B
```

In the next example, enhancements to a core program will be made. Note that there are many versions of BASIC and minor modifications may be necessary to run these programs on particular machines. Every attempt was made to use a most general form in the following.

Example 1. This program generates addition facts with addends in the range 0-10 inclusive. The program must be terminated by program interrupt (e.g., BREAK, control C, RUN/STOP).

```
100  REM SUMS 0-20
120  PRINT "ADDITION DRILL"
140  LET X=INT(11*RND(1))
160  LET Y=INT(11*RND(1))
180  PRINT X;"+";Y;"=";
200  INPUT A
220  IF A=X+Y THEN 280
240  PRINT "WRONG, THE SUM IS ";X+Y
260  GO TO 140
280  PRINT "GOOD"
300  GO TO 140
1000  END
```

Enhancement 1: These modifications give exactly 10 problems and cause automatic termination.

```
Add:     330  FOR C=1 TO 10
```

```
Change:  260 GO TO 300
         300 NEXT C
```

Enhancement 2: Report the number correctly.

```
Add:     10  N = 0
         290  N = N + 1
         320  PRINT "YOU ANSWERED";N;" CORRECTLY"
```

Enhancement 3: Personalize program.

```
Add:     20  DIM N$(30)
         30  PRINT "WHAT IS YOUR NAME"
         40  INPUT N$
         50  PRINT "HELLO ";N$" I HOPE YOU ENJOY THE DRILL"
Change:  320  PRINT N$;" YOU ANSWERED";N;" CORRECTLY."
```

Modified Program:

```
10 N = 0
20  DIM N$(3)
30  PRINT "WHAT IS YOUR NAME"
40  INPUT N$
50  PRINT "HELLO "N$" I HOPE YOU
     ENJOY THIS DRILL.
100  REM  SUMS 0-20
120  PRINT "ADDITION DRILL"
130  FOR C = 1 TO 10
140  X = INT (11*RND(1))
160  Y = INT (11*RND(1))
180  PRINT X"+"Y"=";
200  INPUT A
220  IF A = X + Y THEN 280
240  PRINT "WRONG, THE SUM IS "X + Y"
260  GOTO 300
280  PRINT "GOOD"
290  N = N + 1
300  NEXT C
320  PRINT N$" YOU ANSWERED "N" CORRECTLY"
1000  END
```

Other possible enhancements include:

1 Adding timing functions to limit the time allowed for student response, or control the rate of presentation of

problems.

2 Adding random praise and unpraise routines.

3 Allowing the user to choose the number of exercises.

4 Giving the user a second chance to answer an exercise after answering incorrectly on the first try.

5 Allowing the user to choose the difficulty level.

6 Adding graphics or animation.

7 Adding instruction sequences that are given for particular student responses.

Naturally, each enhancement contributes to the length and complexity of the program and is dependent on the type of machine being used. Some computers are graphics oriented, others have built in timing functions, and the ease of adding enhancements will depend on these idiosyncrasies. The following program will give the user drill in multiplication of integers.

```
50  PRINT "DRILL IN MULTIPLYING
      INTEGERS"
55  PRINT
60  PRINT
100  DIM N$(30)
110 C = S = 0
120  PRINT "WHAT IS YOUR NAME"
130  PRINT
140  INPUT N$
150  PRINT
160  PRINT "HELLO "N$" HOW MANY
      PROBLEMS DO YOU WANT";
170  INPUT N
180  FOR P = 1 TO N
190 X = INT (25 * RND (1)) - 12

200 Y = INT (25 * RND (1)) - 12

210  PRINT
220  PRINT X"*"Y"=";
230  INPUT A
240  IF A = X * Y THEN 350
250  PRINT "WRONG, TRY AGAIN"
260  PRINT
270  PRINT X"*"Y"=";
280  INPUT A
290  IF A = X * Y THEN 320
295  PRINT
300  PRINT X"*"Y"="X * Y
310  GOTO 370
```

```
320  PRINT "OK"
325  PRINT
330 S = S + 1
340  GOTO 370
350  GOSUB 500
360 C = C + 1
370  NEXT P
380  PRINT
390  PRINT N$" YOU HAD "C - 1" CO
     RRECT ON THE FIRST"
392  PRINT  "TRY, "S" ON THE SECOND
     D TRY AND "P - C - S" WRONG.
400  PRINT.
410  PRINT "BYE FOR NOW"
420  GOTO 1000
500  REM  RANDOM PRAISE ROUTINE
510 R = INT(4 * RND (1)) + 1
520  ON R GOTO 550,600,650,700
550  PRINT "GREAT!"
560  GOTO 750
600  PRINT "SUPER!"
610  GOTO 750
650  PRINT "NICE JOB!"
660  GOTO 750
700  PRINT "EXCELLENT!"
750  PRINT
800  RETURN
1000  END
```

Other possible programs include:

1 Given a complex number, enter its conjugate.

2 Estimate the length of a randomly generated line segment.

3 Match mathematics vocabulary words and meanings.

4 Randomly generate three lengths, input whether the three segments can form a triangle. If so, what is the perimeter? the area (at appropriate level)? do the three segments determine a right triangle? an isosceles triangle?

5 Provide practice in estimating products and quotients.

6 Convert units of metric measurement

7 Estimate the ordinate of a point on a graphically represented number line.

References

Hewlett-Packard. SM2, 1966, (computer program).

Kleiman, G. and Humphrey, M. Writing Your Own Software: Authoring Tools Make it Easy. Electronic Learning, May/June, 1982, pp. 36-41.

CHAPTER SEVEN

OTHER USES OF COMPUTERS

Gary Marchionini
Wayne State University
Detroit, Michigan

Al Monetta
Waterford-Mott High School
Waterford, Michigan

Computer Data Bases

A DATA BASE is a collection of symbols stored according to a specific format. The format facilitates retrieval and manipulation of symbols (data) while minimizing the amount of storage space required. A library is an example of a data base in the generic sense. Computers are well suited to storing, sorting and retrieving data, and thus an important use of computers is the maintenance of data bases. Business, industry, and government maintain massive data bases to serve customers or constituents more efficiently. A billing system typically operates on a data base that contains both static and updatable information. A telephone bill, for example, is generated by a billing program which reads a data base containing name, telephone number, address, and the number and type of calls made during the billing period. A posting program automatically updates the data base each time a call is made.

A data base can be used by many different programs or users. It

has been proposed that a huge national data base be constructed which all corporations, individuals, and government agencies share. The Privacy Act of 1974 restricted the consolidation and access of data bases containing information on citizens. Data bases which contain information specific to various professions include ERIC for education, Medline for medicine, and LEXIS for law. Information is fast becoming a mass-marketable commodity. Several general information data bases are available to individuals with microcomputers via the telephone. They provide such information as: airline schedules; financial news, including instant Dow Jones averages; education programs; games; UPI files; weather forecasts; and shopping guides. These rapidly growing services are called INFORMATION UTILITIES. It is likely that information utilities will play an increasingly important role in the lives of students as well as adults.

The two examples which follow illustrate how the ability of the computer to store, manipulate, and retrieve data can be used in the mathematics classroom for the purpose of <u>studying more mathematics</u>. It is important to note that these examples employ the computer in a many-to-one mode--that is a group of students or an entire class can use a single computer.

First, students or a teacher can access an information utility to do research on problems or devise problems, much like a newspaper is used in the mathematics classroom. Valuable life skills can thus be reinforced while pursuing the goals of mathematics instruction. This use of the computer requires a telephone, modem, terminal or microcomputer, and subscription to an information service. Accessing

and interacting with an occupational information system such as the Michigan Occupational Information System (MOIS) is another example of this use of the computer.

Secondly, students can generate data by performing mathematical experiments, or collect data through observational research and use the computer to store and manage the manipulation of the data. Since creating a data base involves file manipulation, only students with extensive programming skills can create their own data base. The teacher may write the necessary software or purchase a data base software package so that students may easily enter and manipulate data. In either case, a time-sharing minicomputer system or floppy disk microcomputer configuration is required. An example of using the computer in this manner follows.

Early in the school year each student is assigned a week during which s/he is to measure the outdoor temperature at the beginning of the class period. The data is entered in the data base and as the school year progresses a collection of real measures is available for analysis or to serve as the basis for practice problems. Students may be asked to construct tables or graphs of the data, predict the temperature based on existing data, or simply operate on the numbers by finding means, modes, and medians for given time intervals. In fact, an entire statistics unit can be integrated into a course in this way. Other skills developed include: Using a thermometer, interacting with a computer data base, estimating results, and predicting outcomes. Students gain paper and pencil as well as electronic mathematical skills. Other data could be collected such as:

body measurements, local traffic volume for given time intervals, daily attendance, or financial figures (e.g., interest rates or Dow Jones averages).

Generation of Data

The computer can be used as a data generator. The computer's ability to generate random numbers (see Chapter Six) makes it a natural device for simulating mathematical experiments (see Chapter Five) or extending data collected by students in an experiment. An example of the latter use is given below.

The study of probability and statistics can be a most enjoyable experience for both teachers and students. The content allows realistic situations to be explored via experiments and activities while providing many opportunities for applying fraction and decimal skills. Repeated coin flipping can provide initial practice in data collection and representation, as well as provide a simple format for introducing concepts such as "sample space", "outcome", and "probability of an event". Experiments with dice follow naturally. Direct groups of students to roll a pair of dice 100 times and record the sums. Efficient collection and organizing techniques should be discussed as well as measures of central tendency and data representations. Consideration of why certain numbers occur more frequently than others, comparison of observed data with expected data, and the various ways of graphically representing data provide a foundation for understanding the limitations of prediction theory as well as the power and relevance of probability and statistics.

After the above procedures have been completed, the computer can be used for extension. Asking students to roll the dice 10000 times should lead to a discussion of large numbers and the infeasibility of conducting such an experiment during valuable class time. Based on generalizations made during the initial experiment with 100 rolls, students should be able to predict the outcomes for 10000 rolls. The program which follows simulates the rolling of a pair of regular dice any number of times. Running the program while students are working on their predictions will provide the basis for discussions of discrepencies between their calculations and the computer's results. The importance of the data is that it is an extension of the actual experiment performed in class and is specific to the individual class.

After consideration of the computer data, an examination of ramdom number generators used by computers could provide enrichment. At the appropriate levels, students could modify the program to simulate the rolling of several dice or the rolling of other polyhedral dice, or compare observed versus expected data in terms of error measurements or Chi square statistics.

The following program executed 10000 rolls on an Apple computer in about five minutes. Although computers are very fast, rolling the dice 1000000 times will drive home the magnitude of 100000 if one has the patience to wait most of a day for the output!

```
100  DIM A(12)
120  PRINT "HOW MANY ROLLS"
140  INPUT R
160  FOR N=1 TO R
180  D1=INT(6*RND(1))+1
200  D2=INT(6*RND(1))+1
220  S=D1+D2
240  A(S)=A(S)+1
260  NEXT N
280  PRINT "FOR ";R;" ROLLS THERE WERE:"
300  PRINT "TWOS", "THREES", "FOURS"
320  PRINT A(2),A(3),A(4)
340  PRINT "FIVES," "SIXES", "SEVENS"
360  PRINT A(5),A(6),A(7)
380  PRINT "EIGHTS", "NINES", "TENS
400  PRINT A(8),A(9),A(10)
420  PRINT "ELEVENS", "TWELVES"
440  PRINT A(11),A(12)
460  END
```

Computer-Managed Instruction

Computer Managed Instruction (CMI) applies the record keeping and data management capabilities of the computer to facilitate individualized instruction. As any teacher who has implemented an individualized program is aware, a great deal of time is needed to maintain individual lessons and records for each student. The data processing powers of the computer are an obvious choice in assisting with the management and analysis of classroom data and coursework.

Since a CMI system maintains individual student records, large curriculum objective banks, and complete catalogs of resource materials by objective, huge amounts of computer memory are required. For this reason, time-sharing computers have generally been used in CMI applications. As storage capacities of microcomputer-based systems are increased, limited micro-based CMI programs are emerging. For

example, one microcomputer-based package uses a card reader to facilitate test scoring.

Teachers who use CMI need little computer experience other than the ability to access the system. Most CMI programs prompt the user, thus enabling easy interaction. The teacher must of course be thoroughly familiar with the curriculum objectives and resource materials and system references.

Ideally, CMI can be used in any content area at any grade level. Several large scale CMI programs have been developed for mathematics education. Some of the better documented programs include: The Cincinnati Instructional Management System (CIMMS), various PLATO programs such as the Basic Skills Learning System, and the Michigan Teacher Support System (MTSS).

Most individualized instruction programs include: diagnosis, prescription and/or instruction, evaluation and feedback. The key to designing a CMI system is the identification of objectives, and the correlation of them to the kits, texts, films, and concrete aids available in particular classrooms. Generally, this important process is carried out by teams of curriculum experts, teachers and computer specialists. Once the program is to be implemented in a classroom, a terminal is provided for the teacher or building and pick-up service for paperwork flow is established.

The teacher administers a general inventory and the test papers are sent to a central location for machine scoring. The teacher

accesses the system when scoring is complete and receives a printout of student performance by content area, and a battery of specific diagnostic tests for each area are assigned for each student as needed. Based on the outcomes on the individualized diagnostic tests, specific prescriptions are printed for each student. A student may be directed to a textbook lesson or a kit that provides instruction or practice on particular objectives. The teacher can access various reports on individual students, the entire class, various objectives or content areas. If students are moving too quickly or too slowly through a unit, the teacher is alerted and appropriate adjustments to the student's prescriptions or set of objectives can be made with a few simple commands. Upon completion of a set of objectives, students are assigned post tests and, based on the results, appropriate remedial prescriptions made or the next diagnostic test assigned. Once students and the teacher become familiar with the CMI routine the typical classroom scene includes: some students taking tests, others working individually on prescriptions and others receiving small group or individual instruction by the teacher.

It is important to note that students do not use the computer, but rather the teacher can use CMI with only one terminal. For this reason, CMI promises to be a cost effective use of the computer once initial system development is complete. Corporations, the armed services, and some schools have used CMI with positive results. Although the computer can aid in the individualized classroom, an energetic, well organized teacher is a prerequisite. Interacting with multiple groups or individuals on a variety of topics while proctoring tests and managing the minute to minute interactions in a typical

classroom requires stamina, patience and dedication. CMI does not promise to make teaching easier, but it does offer the chance for making it more effective.

Computer Generation of Worksheets and Tests

Computer programs can be designed to generate practice worksheets or tests. The computer generates the worksheets or tests according to detailed specifications given by the teacher. Each student in a class can be given a worksheet or test customized according to particular strands or objectives. The teacher can also use the computer to generate a group test or worksheet that includes any number of items on any number of objectives.

Comprehensive worksheet or test generating programs are available on time- sharing systems which can store the large objective banks required. Drill and practice programs like those discussed in Chapter Six can be modified to print the problems on a printer rather than waiting for input and interacting with the student. A necessity for using this application is the availability of a printing terminal. Since a large amount of printing is required daily, a high speed printer is the ideal device. The investment is quickly amortized since a single terminal or microcomputer can serve an entire school.

One package that addresses K-8 computational skills is the Minnesota Educational Computing Consortium's (MECC) COMPUTE. Available on several time-sharing systems, COMPUTE will allow the teacher to specify which and how many of 459 objectives should be included on a worksheet. The topic areas include: whole number operations, fractions, decimals, money, and percent. The program is interactive

and prompts the teacher for objective codes and number of items. Individual or group worksheets or tests are printed with appropriate answer keys.

Computer generated tests and worksheet programs can be used in individualized classrooms or to provide supplementary resources for any instructional setting. As with CMI, large investments of time must be made in identifying and establishing objective banks. Presently, most applications include computational objectives since they are easy to write behaviorally and large numbers of individual problems can be randomly generated by specifying parameters within the program. As with drill and practice programs, the computer provides no instruction but only supplementary practice, useful after concepts have been developed.

Future Applications of Computers

Future applications of computers in education are only limited by the imagination of educators and the development of appropriate hardware and software. It seems certain that technological advances will continue to rapidly outstrip software developments. Some forces that are likely to affect education include:

Video Disk Technology - the capability to store massive amounts of data digitally makes the video disk a natural storage medium for digital computers. Today a full length film, or the entire set of Encyclopedia Britanica can be stored on a single $5 video disk.

Cable T. V. Expansion - as cable television becomes increasingly

commonplace in homes, the facility for two way interaction between people in separate locations as well as the availability of multiple educational programs may change the school environment significantly. The term "homework" may take on entirely new meanings.

Home Computers - entering the home initially for entertainment purposes, educational programs for use at home are already appearing on the market. The home computer will eventually control the myriad of other technologies available in the home.

Phone Information Services - many services such as sports information, weather reports, etc., have long been available via a telephone call. As information libraries become publicly available, it is likely that entire computerized libraries of information will be available by making a single phone call.

Voice Input and Output capabilities may revolutionize computer use. Voice synthesizers a la Speak and Spell are already providing exciting prospects for machine-directed oral learning. Voice input is technologically less developed, but the possibilities of programming or commanding a computer by speaking to it lie somewhere between the fantastic and science fiction.

A few software developments have greatly enhanced the use of computers by people with little computer training. These include: word processors, interactive data managing programs (e.g., Visicalc), authoring languages, and various programmer aids. As software advances continue, it is likely that just as modern kitchens are equipped with a wide assortment of machines to perform particular tasks, the homes

and offices of the future will be equipped with a wide array of software packages to perform particular intellectual tasks.

Today's electronic bulletin boards and message services allow users in all parts of the world to communicate instantly. PLATONOTES, CONFER, and other computer user networks are the forerunners of future electronic mail systems.

The research taking place in the field of artificial intelligence promises not only to improve the capabilities of machines but also to continue to provide insight into the problem solving and other higher cognitive processes of human beings.

CHAPTER EIGHT

RESOURCES FOR EDUCATIONAL COMPUTING

Wayne Kladder
Traverse City Jr. High
Traverse City, Michigan

Resources in this chapter are meant to be representative rather than exhaustive. Selected magazines and journals, books, and organizations are listed. Evaluation criteria for software selection are also included. Note that NCTM has published software evaluation criteria and forms (see Guidelines for Evaluating Computerized Instructional Materials.

Resource Projects

MINNESOTA EDUCATIONAL COMPUTING CONSORTIUM (MECC)
MECC is a statewide organization of educational institutions which coordinates computing activities of members systems and serves those systems by supplying hardware and software at wholesale rates, and training and consultation. MECC markets educational software at reasonable rates.

Contact: MECC
2520 Broadway Dr.
St. Paul, Minnesota 55113
(612) 376-1122

TABS - PROJECT
TABS is an Education Department funded project at the College of Education at Ohio State University. The purpose of the project is to develop, collect and disseminate exemplary mathematics curricular materials in which computers and other information processing technology are used.

Contact: Suzanne K. Damarin
TABS Project
ARPS Hall 202-A
1945 N. High St.
Columbus, Ohio 43210

MICROSIFT
MICROSIFT is an NSF-funded project located at the Northwest Regional Educational Laboratory, MICROSIFT's purpose is to act as a clearinghouse for microcumputer K-12 instructional software and information. Member organizations will collect, evaluate and disseminate instructional software.

Contact: Don Holznagel
MICROSIFT
500 Lindsay Bldg.
710 S.W. Second Ave.
Portland, OR 97204

REGION IV EDUCATION SERVICE CENTER
Region IV is one of twenty state-funded regional organizations in Texas. The center offers a wide range of support to member schools. Region IV provides computer support by supplying training, wholesale hardware rates, and participates in the MICROSIFT software evaluation project.

Contact: Pat Sturdivant
Region IV Education Service Center
P. O. Box 863
1750 Seamist St.
Houston, TX 77001

Tri-County Computer Consortium (TCC)
Co-sponsored by Oakland, Wayne, and Macomb Intermediate School Districts, TCC will collect, evaluate, and disseminate public domain educational software for the Apple, Atari, PET and TRS-80 microcumputers. The consortium has also produced a Computer Literacy Curriculum Guide (Project CLAS).

Contact:

T. Schmeltzer	T. Hartsig	L. Smith
Oakland Schools	Macomb ISD	Wayne ISD
2100 Pontiac Lake	44001 Garfield	33500 Van Born
Pontiac, MI 48054	Mt. Clemens, MI 48043	Wayne, MI 48184

SOFTSWAP
Co-sponsored by the San Mateo County Office of Education and Computer-Using Educators (CUE), SOFTSWAP collects and disseminates software for Apple, PET, and Radio Shack microcomputers.
Contact: SOFTSWAP
333 Main Street
Redwood City, CA 94603

Human Resources Research Organization (HumRRO)
Sponsored by a NSF grant, HumRRO and the Montgomery County (Maryland) Public Schools have produced a Computer Literacy K-8 Curriculum Guide.

Contact: Beverly Hunter
HumRRO
300 N. Washington St.
Alexandria, VA 22314

Professional Organizations

Association for Computing Machinery (ACM)
ACM has several special interest groups that focus on particular aspects of computing. The Special Interest Group on Computer Science Education (SIGCSE) and Special Interest Group on Computer Uses in Education (SIGCUE) produce periodic reports relative to educational applications of computers.
Contact: ACM
1133 Avenue of the Americas
New York, NY 10036

Association for Educational Data Systems (AEDS)
AEDS is a professional organization for educators interested in the application of computers and technology in teaching and learning. A journal, monographs, a newsletter and conferences are benefits of membership.

Contact: AEDS
1201 Sixteenth St. N.W.
Washington, D.C. 20036

International Council for Computers in Education (ICCE)
ICCE is an international organization for educators interested in instructional uses of computers at the precollege level. The organization publishes The Computing Teacher.

Contact: David Moursund
Dept. of Computer and Information Science
University of Oregon
Eugene, OR 97403

Michigan Association for Computer Users in Learning (MACUL)
MACUL is an organization of educators interested in the educational applications of computers. A newsletter, a journal, and several conferences per year are sponsored by MACUL.

Contact: MACUL

c/o Lary Smith
33500 Van Born Rd.
Wayne, MI 48184

Newsletters, Journals and Magazines

Classroom Computer News

Classroom Computer News
Box 266
Cambridge, MA 02138

Creative Computing

Creative Computing
P. O. Box 789-M
Morristown, NJ 07960

Electronic Learning

Scholastic Inc.
902 Sylvan Avenue
Box 2001
Engelwood Cliffs, NJ 07632

Personal Computing

Personal Computing
1050 Commonwealth
Boston, MA 02215

Recreational Computing

Recreational Computing
Dept. T7
P. O. Box E
Menlo Park, CA 94025

The Computing Teacher

The Computing Teacher
Dept. of Computer and
Information Science
University of Oregon
Eugene, OR 97403

TRS-80 USERS

CLOAD Magazine
P. O. Box 1267
Goleta, CA 93017

80-U.S. Journal
3838 S. Warner St.
Tacoma, WA 98409

TRS-80 Microcomputer News
700 One Tandy Center
Fort Worth, TX 76102

Softside
P. O. Box 68
Milford, NH 03055

APPLE USERS

Call A.P.P.L.E.
304 Main Ave. S.
Suite 300
Renton, WA 98055

Apple Orchard
International Apple Corps.
P. O. Box 976
Daly City, CA 95017

Nibble
P. O. Box 325
Lincoln, MA 01773

Softalk
Softalk Publishing Inc.
10432 Burbank Blvd.
N. Hollywood, CA 91601

PET USERS

Cursor
Box 550
Goleta, CA 93017

PET User Notes
PET User Group
P. O. Box 371

Montgomeryville, PA 18936

Books and Reports

Are You Computer Literate? Billings and Morsund. dilithium press. This book gives a low-technical introduction to computers and literacy.

BASIC and the Personal Computer. T. Dwyer and M. Critchfield. Addison-Wesley. This book gives a thorough introduction to BASIC and also covers graphics, data files, simulation, sorting, art, games and word processing.

BASIC Computer Games and More BASIC Computer Games. D. Ahl (ed.) Creative Computing Press. Each book contains many games with full listings and sample runs. Students gain an understanding of the computer while keying in the games.

BASIC Programming. Kemmeny and Kurtz. Wiley Publications. This book begins with a simple introduction to BASIC and proceeds to applications, including files, games, text, graphs, and mathematics.

Computer Alphabet Book. E. Wall. Bayshore Books. This book presents the "ABC's of computer awareness.

Computer Consciousness Surviving the Automated 80s. D. Covvey and N. H. McAlister. Addison-Wesley. A humorous and understandable introduction to computers in our world.

Computer Graphics Primer. M. Waite An introduction to computer graphics on the Apple II microcomputer.

Computers for Kids. S. Larsen. Creative Computing Press. Introduces computers to elementary school students. Editions available for Apple, Pet, and TRS-80 computers.

Computers in Mathematics: A Sourcebook of Ideas. D. Ahl (ed). Creative Computing Press. A book of ideas for using the computers in mathematics classes.

Computers in the School: Tutor, Tool, Tutee. R. Taylor (ed.). Teachers College Press. Collection of articles by A. Bork, T. Dwyer, A. Luehrmann, S. Papert, and P. Suppes. Good overview of educational applications of computers.

Instant BASIC. J. Brown. Dymax Publishers. This book gives an interactive, non-mathematical means of learning BASIC.

Guide to Microcomputers. F. Frederick. Association for Educational Communications and Technology, 1980. Gives a good overview of various microcomputers, sources for information and glossary.

Guidelines for Evaluation Computerized Instructional Materials. NCTM.

Suggestions and procedures for evaluating and selecting software are given.

Katie and the Computer. F. D'Ignazio and S. Gilliam. Creative Computing Press. Students learn how a computer works as Katie goes on an adventure through the inside of a computer.

Learning with Computers. A. Bork. Digital Press.
A book that includes plenty of "how to" information for designing and using computer courseware.

Microcomputers and the 3 R's. C. Doerr. Hayden, 1979.
Many examples of computer applications in science, mathematics, language arts, and social studies are given.

Microcomputers for Teachers--With Application to Mathematics and Science. D.Zalewski (ed). School Science and Mathematics Association. A collection of ways to use microcomputers in the mathematics and science classroom.

Mindstorms. S. Papert. Basic Books.
Advocates the use of LOGO, which allows the student to program the computer, not the computer programming the child.

My Friend, the Computer. J. Rice. Denison and Company, 1976.
Elementary introduction to computers and computer programming.

Problem Solving With the Computer. T. Gage. McGraw.
This book stresses problem analysis in traditional algebra and geometry classes.

Stimulating Simulations. C. Engel. Hayden
A book of twelve games with complete listings, sample runs and documentation. A sourcebook for programming projects.

Teacher's Guide to Computers in the Elementary Schools. D. Moursund. ICCE, Computing Center Eastern Oregon State College. This booklet is designed to help elementary teachers gain an initial level of computer literacy and to lay a foundation for additional learning.

Technology in Science Education: The Next Ten Years. D. Deringer and A. Molnar. National Science Foundation. Four articles dealing with computer applications in the 1980's.

The ENTELEK Computer-Based Math Lab. L. Stolurow. ENTELEK.
Contains 35 CAI programs in BASIC for use in high school mathematics classes. The programs are keyed to secondary math texts. Some topics are: estimating square roots, logarithms of various bases, roots of equations, and trig. functions.

TOPICS: Computer Education for Elementary and Secondary Schools.
Association for Computing Machinery. A summary of the work of the ACM Elementary and Secondary Subcommittee. Good source of information for schools learning about using computers.

1970-1980 Microcumputing Report. Recommendations for schools as they begin the process of selecting microcomputers. Minnesota Educational Computing Consortium.

Selected Articles

Bell, F. Can Computers Really Improve School Mathematics? The Mathematics Teacher, 1978, 71, pp. 428-433.

Carlson, R. Buffon's Needle Problem on a Microcomputer. The Mathematics Teacher, 1981, 74, pp. 638-640.

Dorn, W. The Computer: A Mathematics Laboratory. Journal of Educational Data Processing, 1968, 5, pp. 67-79.

Gesshel-Green, H. Getting Started in a High School. The Mathematics Teacher, 1981, 74, pp. 610-612.

Hatfield, L. A Case and Techniques for Computers: Using Computers in Middle School Mathematics. The Arithmetic Teacher, February, 1979, pp. 53-55.

Hutcheson, J. Computer-Assisted Instruction is not Always Drill. The Mathematics Teacher, 1980, 73, pp. 689-716.

Inman, D. and Clyde, D. Computer Applications. The Mathematics Teacher, 1981, 74, pp. 618-620.

Landry, M. Algebra and the Computer. The Mathematics Teacher, 1980, 73, pp. 663-667.

Levin, D. Microcomputers: Out of the Toy Chest and Into the Classroom. The Executive Educator, March 1980, pp. 19-25.

Robbins, B. and Taylor, R. Getting Started in a Junior High School. The Mathematics Teacher, 1981, 74, pp. 605-608.

Snover, S. and Spikell, M. Problem Solving and Programming: the License Plate Curiosity. The Mathematics Teacher, 1981, 74, pp. 616-617.

SOFTWARE EVALUATION FORM

Name of program_____
Cost_____Format (disk, tape, etc._____
Type of computer required_____
Memory required_____Vendor_____

1. Abstract

2. In what specific course(s) might this program be used? (for example: mathematics, reading, chemistry, etc.)

3. What are the educational objectives to be met via this program?

4. Indicate an appropriate grade level for the program.

Pre-school_____ Junior High_____
Lower Elem_____ Senior High_____
Upper Elem_____ College_____

5. Indicate the prerequisite skills necessary to use this program.

6. Indicate the type(s) of techniques used in the program.

Drill and practice_____ Simulation_____
Educational game_____ Teacher aid_____
Testing_____ Problem sloving_____
Tutorial_____ other_____

7. Is it appropriate to use the computer to teach/learn this material (or is there a more effective method)?

_____ yes _____ no _____not sure _____ N/A

8. What degree of teacher experience with computers is necessary? Why?

_____ great deal _____some _____little _____none

9. What motivational devices are used in the program?

	YES	NO	NOT SURE
Timing			
Scoring			
Personalization			
Graphics			
Sound			
Color			
Other (specify)			

10. Identify the strengths of this program.

11. Identify the weaknesses of this program.

12. What protection does the vendor provide for backup if the program is damaged or does not operate properly?

13. Summary Checklist

	Strongly agree	Agree	Disagree	Strongly disagree	Not applicable
Has educational value					
Content is accurate					
Easy to use					
Instructions are good					
High interest level					
Includes written documentation					
Accomplishes its objectives					
Good use of graphics, color, and sound					

14. Do you recommend purchase of this program? Why?

15. Comments: